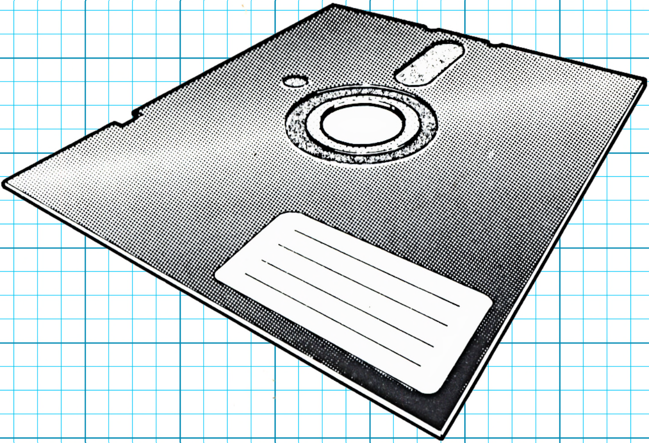
An Introduction to CP/M

R.A. PENFOLD



ALSO COVERS CP/M PLUS

AN INTRODUCTION TO CP/M

OTHER BOOKS OF INTEREST

BP187 A Practical Reference Guide to Word Processing on the Amstrad PCW 8256 and PCW 8512
BP188 Getting Started with BASIC and LOGO on the Amstrad PCW 8256 and PCW 8512
BP189 Using Your Amstrad CPC Disc Drives

AN INTRODUCTION TO CP/M

by

R. A. PENFOLD

BERNARD BABANI (publishing) LTD
THE GRAMPIANS
SHEPHERDS BUSH ROAD
LONDON W6 7NF
ENGLAND

PLEASE NOTE

Although every care has been taken with the production of this book to ensure that any projects, designs, modifications and/or programs etc. contained herein, operate in a correct and safe manner and also that any components specified are normally available in Great Britain, the Publishers do not accept responsibility in any way for the failure, including fault in design, of any project, design, modification or program to work correctly or to cause damage to any other equipment that it may be connected to or used in conjunction with, or in respect of any other damage or injury that may be so caused, nor do the Publishers accept responsibility in any way for the failure to obtain specified components.

Notice is also given that if equipment that is still under warranty is modified in any way or used or connected with home-built equipment then that warranty may be void.

CP/M is a registered trade mark of Digital Research

© 1986 BERNARD BABANI (publishing) LTD

First Published – October 1986

British Library Cataloguing in Publication Data
Penfold, R.A.
An introduction to CP/M
1. CP/M (Computer operation system)
2. Microcomputers
I. Title
005.4'46 QA76.6

ISBN 0 85934 157 7

Printed and Bound in Great Britain by Cox & Wyman Ltd, Reading

Preface

One of the first things that becomes apparent to anyone who becomes involved in computing is the problem of software compatibility. There are a multitude of different computers, each requiring their own version of a program before they can load and run it properly. There have been attempts to overcome this problem, and CP/M is one of the most successful of these. There are almost countless computers that are capable of running under the CP/M operating system, and the CP/M system effectively irons out differences between computers so that they can run the same software (provided it is designed to operate with a CP/M system of course). The success of CP/M has been such that there are now thousands of programs available which run under this system.

CP/M is more than just a program to give a common set of standards, and hence software compatibility between various machines. It includes a range of commands that help with such things as file copying, file editing, and directing data output to the appropriate device. In order to use and run applications programs running under CP/M it is not essential to have even a basic understanding of this system, but a reasonable knowledge of the subject can certainly be of immense help when minor problems occur, and also helps the user to fully exploit the potential of the system. This book gives details of the CP/M commands, and it does not assume a great deal of technical knowledge on the part of the reader. It does assume that the reader has access to a computer which will operate under CP/M, so that some simple instructions which help to demonstrate the way in which commands operate can be entered and run. It is not absolutely essential to try out the example instructions, but doing so greatly adds familiarisation with CP/M commands and makes it much easier to gain a proper understanding of the way in which they function.

R. A. Penfold

CONTENTS

Page

Chapter 1

CP/M BASICS 1
Discs and Drives 5
Formats 11
Other Disc Sizes 13
Hard Discs 14
Getting Started 15
DIRectory 17
ERAse 19
REName 28
STAT 29
SET 33
Printer 35

Chapter 2

USING PIP 36
PIP Command 36
PIP as a Program 38
Multiple Files 39
SYSGEN 41
Devices 43
Screen Layout 50
Data Transfers 52
Concatenation 56
Partial Copies 58
Finally 60

Chapter 3

SET AND USERS 61
Passwords 61
READ 62
WRITE 62
DELETE 62
NONE 62
LABELS 65
USER 66
ASM 67
GET 70
PUT 73
SUBMIT 74
Finally 76

Page

Appendix 1
CONSOLE AND ED COMMAND CODES 77

Appendix 2
MAIN ED COMMANDS 78

Appendix 3
PIP PARAMETER OPTIONS 80

Appendix 4
CP/M COMMANDS 82

Appendix 5
EXTENSION TYPES 83

Chapter 1

CP/M BASICS

For the newcomer to computing, or even for someone who has some experience with computers, it can be a little difficult at first to understand the basic function of an operating system such as CP/M. It is, of course, a program which is loaded into the computer, and it will respond to certain commands. In this respect it resembles a programming language such as BASIC or LOGO, but it is something less than a full computer language, and although you will often come across references to CP/M programs, these are not written in CP/M in the same way that programs are written in BASIC or some other computer language. The programs are written in machine code, BASIC, or some other computer language, and run under the CP/M operating system.

In order to understand what CP/M is all about, and what makes it so useful, it is perhaps best to first consider operating systems in general. All computers have some form of built in operating system, however crude, and without one the machine would simply crash at switch-on and would be useless. The operating system is a program contained within the computer in a type of memory known as ROM (read only memory), and as its name suggests, data and instructions can only be read from ROM, and the contents can not be changed once the device has been programmed at the manufacturing stage. This is in fact an advantage for something like an operating system where it avoids the possibility of accidentally overwriting the ROM contents and destroying the vital program it contains. Perhaps of more importance, ROM is a non-volatile form of memory, which simply means that its contents are not lost when the computer is switched off.

With many home computers the operating system does little more than perform some simple setting up routines and then hand over control to a BASIC interpreter or some other form of programming language. With other computers the operating system is quite complex, and although control may be handed over to a high level programming language almost immediately at switch-on, the operating system can still be a large program which is almost continuously being accessed by the high level language.

The purpose of the operating system is primarily to enable the various peripheral elements of the set up to work together as a proper and co-ordinated system, rather than in a haphazard and disorganised way. This is essential if the computer is to perform even quite simple tasks efficiently and without crashing. It has to be emphasized that by "peripheral elements", I do not simply mean the usual peripheral devices such as disc drives and printers, but anything to which the computer provides an output or receives any input. In other words, it includes things such as the keyboard and the monitor screen which would ordinarily be thought of as integral parts of the computer rather than peripherals. The reason for including these as peripherals is simply that to the microprocessor at the heart of the computer they are all input/output devices, and are treated in much the same way regardless of whether they are built into the main unit or are add-ons of some kind.

Where a computer has a complex operating system this can be of advantage when running an application program or a programming language other than any built in language. For example, the programming language or application program may well need to scan the keyboard at frequent intervals. If the operating system has a routine for this scanning then the program running in the computer can simply jump to this routine each time a scan is required, rather than having to provide its own routine. A full operating system provides routines to control all the peripheral elements in the system, making it easy to write data to the monitor screen, read it from a serial port, or whatever.

In an ideal world there would be a high degree of comptibility between various computers aimed at particular types of user, so that programs written for one machine could easily be changed to run on similar types. In practice this ideal is a pipedream, and computers tend to have totally different operating systems, making it a relatively difficult and time consuming task to rewrite a program for one computer to run on a different type. One problem is simply that the hardware present varies from one machine to another, both in terms of what peripherals are actually included in the system, and the particular pieces of electronics which are used to generate the screen display and so on.

CP/M is a registered trademark of Digital Research, and it stands for "Control Program for Microprocessors". The basic idea is to provide a standard operating system for a wide range of different computers, so that (with certain reservations) a program written for one machine running CP/M will operate on any other CP/M computer with little or no alterations required. The CP/M operating system program itself is different for each computer, as it has to be customised to suit the particular hardware present in each machine. You therefore need to have the correct version of CP/M for your make and model of computer before it can be loaded and used to run CP/M programs.

Although CP/M is by no means available for every computer ever made, it is applicable to a surprisingly wide range of computers. It is mainly associated with computers that are based on the 8080A or (more commonly) the Z80 microprocessor, but it can run on machines that are based on other types of microprocessor (although this usually involves the use of some extra hardware in the form of a Z80 second processor rather than a solution fully implemented in software). The limitations on CP/M compatibility between various machines are inevitable due to a general lack of standardisation in the computer world, as well as the inevitable differences in the sophistication of different computer systems. It is obviously no good trying to load a CP/M program if it is on a disc that is incompatible with the disc drives of your computer. Also, it is not possible to use a program properly if it requires the system to include a printer, serial port, or some other item of hardware which your system does not include, or if your computer simply lacks the required memory capacity.

CP/M is often called a "disc operating system", which has led to the popular misconception that it is only concerned with the control and operation of the disc drives in the system. This is not the case, and while it is true that the disc drives are very much at the centre of things as far as most users are concerned, a full range of peripherals are controlled by CP/M. It is a disc operating system in the sense that it is not built into the computer in the form of a program on ROM, but must be loaded from disc by the user. The exact procedure for loading CP/M varies from one computer to another, and it depends on whether the computer is designed specifically to run CP/M or whether CP/M is just one of a range of options. In most cases these days computers are not designed specifically for CP/M operation, and can usually run some machine-specific software as well as CP/M utilities and applications. For example, I use the popular Amstrad CPC6128 with a word processor and spelling checker which run under the built-in operating system, and a drawing program which runs under CP/M. The manual for your computer or a separate CP/M manual should give details of how to load CP/M, and this is unlikely to require more than placing the program disc into drive "A" and typing a simple command into the computer. For instance, with the Amstrad CPC6128 the command | CPM is used to load the CP/M operating system.

While it is possible to run and use CP/M applications programs without having even a basic understanding of the CP/M system, as with most things associated with computing, an understanding of the subject can be very useful. Not only can it help you to make more effective use of the computer system, but it can also help to get you out of difficulties if things should go slightly wrong.

An unfortunate aspect of CP/M is that there are several versions, and although this slightly complicates things, it is understandable in that the original version has had to be repeatedly changed in an attempt to keep pace with the continuous advances in computer technology. When CP/M came into being, a computer with twin disc drives and 64k of RAM was rightly regarded as an expensive business system. These days such a set up tends to be regarded as rather basic, and many home computer users have machines fitted with twin disc drives and 128k or more of RAM.

The two most common forms of CP/M are CP/M 2.2, and CP/M 3 (which is more commonly called CP/M Plus). The 2.2 version is the final one (at the time of writing this book anyway) for a computer with the standard 64k of RAM. This is the maximum amount of RAM that the 8080 and Z80 microprocessors can directly address. However, some computers based on these microprocessors but having much more than 64k RAM have appeared, and this is achieved by some form of memory bank switching. In other words, the memory is organised in blocks of (say) 64k, and the microprocessor can only access one block of RAM at a time. Some form of electronic switching enables it to select any desired RAM bank though, so that all the RAM can be accessed. This is somewhat slower and less convenient than with a 16 bit microprocessor that can directly address a few megabytes of RAM (1000k = 1 megabyte), but on the other hand it greatly increases the scope of an 8 bit type such as the Z80.

CP/M 2.2 can be used on a computer which has bank switched memory, but as only the basic 64k will be recognised by the operating system, it will considerably less than fully utilize the computer's resources. The main difference between CP/M 2.2 and CP/M Plus is that CP/M Plus is designed to be able to use bank switched memory, making it ideal for the many 8 bit machines which now sport this feature. It also has one or two other refinements, but CP/M 2.2 and CP/M Plus are basically the same. Most of the information provided in this book therefore applies to both these versions of CP/M (as well as most earlier versions come to that), and where there are divergences between the two versions these will be pointed out.

An important point to note is that applications programs can not be guaranteed to run properly unless you are running the right version of CP/M, and obviously CP/M Plus can only run on a computer which supports bank switched memory.

Discs and Drives

CP/M is perhaps a little easier to understand if you are familiar with the fundamentals of discs and disc drives. Being a disc based operating system it is a requirement of any computer system running under CP/M that it has at least one disc drive, and it is more usual for there to be two drives. Most disc drives are for the so called "floppy" discs, and the standard size of floppy disc is eight inches. However, the smaller (5¼ inch) mini floppy discs, or "diskettes" as they are sometimes called, are now much more common. Also, some smaller types (3 inch and 3½ inch) are now coming into use and rapidly gaining in popularity.

A floppy disc system could reasonably be regarded as a cross between a magnetic tape system and a record player. The discs resemble ordinary gramophone records, but are much thinner and are not rigid (hence the term "floppy" disc). The discs do not have grooves like a gramophone record, but are instead coated with a magnetic coating of the type used on magnetic recording tape. The disc drive has a recording/playback head which can be moved across the rotating disc In this way it is possible to record data onto the disc and to recall it again, but data is not placed onto the disc in a continuous spiral like an ordinary gramophone record. Instead the head has a number of fixed positions, with each one covering a track around the disc. Each track is subdivided into several sectors. A typical arrangement is shown in Figure 1.

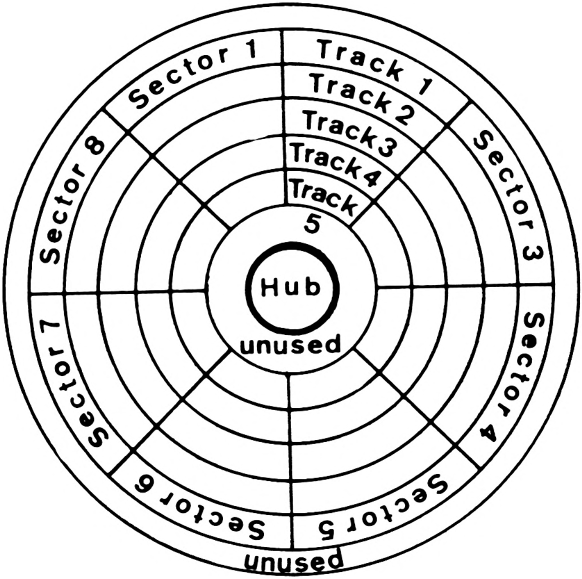

Sector 1
Track 1
Track 2
Track 3
Track 4
Track 5
Sector 8
Sector 3
Hub
unused
Sector 7
Sector 4
Sector 6
Sector 5
unused

Fig. 1. Information is stored on a disc in blocks using this track and sector arrangement. In practice there are many more blocks than shown here.

This fragmented approach may not be very good for ordinary audio recording, but for digital storage it is ideal. The data is recorded onto the disc in blocks, with each block occupying one sector of the disc. The main advantage of disc systems over a more fundamental system such as program storage on cassette tapes is the random access that a disc system provides. In other words, the recording/playback head can be moved almost instantly to any track and sector of the disc, and it can therefore jump almost instantly to the beginning of any program stored on the disc.

Of course, discs are not only suitable for program storage, and they can also be used for such things as storing wordprocessor documents. In fact it is in this type of application that their use is most advantagous, since it is often necessary to store numerous short data files, and with a disc system any file can be almost instantly accessed. It is file rather than program storage which is likely to be of most interest to the majority of CP/M users, and CP/M makes it quite easy to perform tasks such a deleting and copying files.

Obviously a disc system has to incorporate some means of enabling the head to jump straight to the required file rather than simply going through the disc track by track and sector by sector until it finds the right data. Otherwise the random access capability would not be utilized and a disc system would have relatively little advantage over a lower cost method such as a cassette tape storage system. There would still be some advantage in that the slowest disc systems store and retrieve data at a rate which is comparable to the fastest cassette system , but generally with much better reliability. The fastest disc systems are several times faster than any cassette system and still achieve excellent reliability.

The way in which a disc system directs the head to the correct track and sector is by allocating part of the disc to act as a directory. This holds the name of each file on the disc plus its position on the disc. When you call up a file the disc operating system reads through the directory until it finds the right file name, and then it reads the start position of the file. It can then jump straight to ths correct track and sector, and start reading the file. The directory is also used when placing files onto the disc, as it can be read to find suitable free areas of the disc where the file can be recorded. This is normally done by the disc operating system, and is not something with which the user would usually become closely involved. It is possible to inadvertently erase or record over files that are wanted, but with most disc systems (including CP/M) this is unlikely to occur unless the user gets really careless.

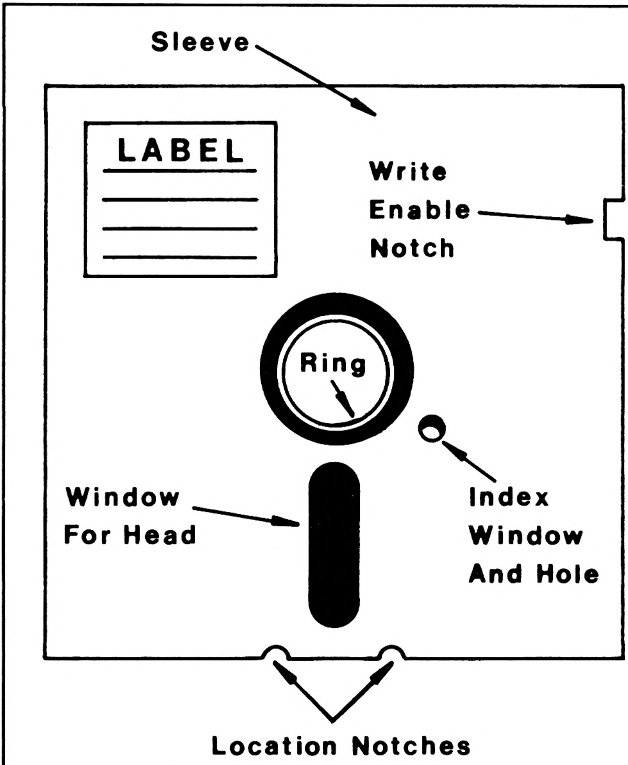

Sleeve
LABEL
Write
Enable
Notch
Ring
Window
For Head
Index
Window
And Hole
Location Notches

Fig. 2. External appearance of a 5¼″ floppy disc.

Figure 2 illustrates the basic appearance of a 5¼″ inch floppy disc, and you will notice that there is a write enable notch in one edge of the disc. This is similar to the tabs on the rear of a cassette which can be removed in order to prevent the contents of the cassette from being over-written. In this case though the notch is normally exposed and it must be covered over with a tab (or insulation tape or something of this nature) in order to protect the contents of the disc.

One thing that often puzzles newcomers to computing is the fact that floppy discs are square rather than round. In fact the disc proper is round, but it is contained in square protective sleeve. This is not like an ordinary gramophone record where the disc is removed from the sleeve prior to being placed on the player. Floppy discs are placed in the disc drive still in the protective sleeve. A slot in the sleeve enables the head to come into contact with the disc. Discs are supplied in an outer protective sleeve which must be removed before the disc is inserted into the player. Discs should always be stored in the outer protective sleeve and the exposed surface of the disc that can be seen through the head window should never be touched. Finger prints can attract dust which can in turn ruin the disc and cause at least part of the data it contains to be lost. Disc systems are generally very reliable, but it is still good practice to record any important files twice, and on separate discs. It is also a good idea to store the two sets of discs separately.

Another aspect of floppy discs which puzzles many people is the index window. This operates in conjunction with the index hole, which is a small hole in the disc itself. If you examine a disc, by placing two fingers into the centre hole and slowly rotating the disc you can carefully align the index hole so that it can be seen through the index window. A photoelectric circuit in the disc drive is used to detect when the index hole and window are aligned, and this provides essential timing information for the disc drive's control system. Remember that with the track and sector system it is necessary for the disc control system to be able to place the head at the correct position on the disc in two planes, and not just on the required track.

The two notches on one edge of the disc are to help with the correct positioning of the disc within the disc drive. When the disc is placed into the drive these should be facing forward. The label enables the disc number or some other means of identification to be marked onto the disc. However, great care must be exercised when doing this as too much pressure could seriously damage the disc. A soft pencil or fibre-tipped pen are ideal for writing identification labels onto discs.

A point that is worth bearing in mind when using cassettes, floppy discs, or any form of magnetic recording medium, is that strong magnetic fields can erase or damage the recording. Even if considerably less than full erasure or only slight damage occurs it could still result in valuable files being unloadable. Cassettes and floppy discs should not be stored near large loudspeakers or anything else which contains a powerful magnet. Also, they should not be stored anywhere that is likely to get quite hot. Bear in mind that even direct sunlight can produce quite high temperatures. For most computers there are special programs available which can be used to rescue at least some of the data on a spoilt disc, but it is much better to never reach the stage where one of these is needed.

8 inch discs are very similar in appearance to the 5¼ inch type, but they have two index holes/windows. Of more importance the system of write-protecting is different, with the tab normally being in place, and being removed to prevent data from being written to the disc.

3 inch and 3½ inch diskettes have strong similarities to the 5¼ inch variety, but there are some important differences. Figure 3 shows the general appearance of a 3 inch diskette. The most striking difference is that the disc is contained in a rigid plastic casing, which is in turn generally housed in a rigid plastic case during storage. This makes the disc far less vulnerable to physical damage, and is probably well worth any increase in cost that it causes. Also, there is a shutter mechanism which covers over the head and index windows when the disc is not in use, so that there is no need to worry about accidentally touching the surface of the disc itself. Again, this is probably well worth the additional expense it imposes. Note though, that although the disc is mechanically well protected it still needs to be treated with some respect, and it is still vulnerable to strong magnetic fields.

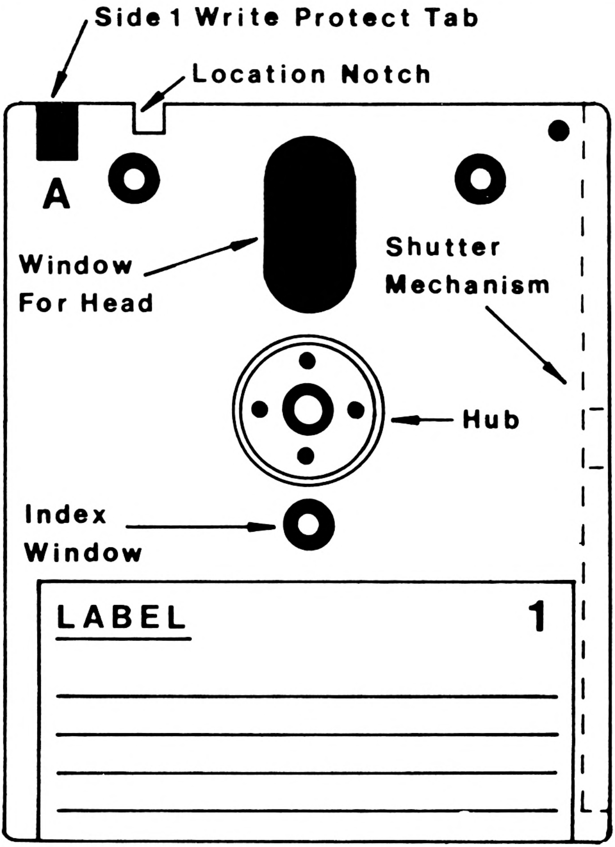

Side 1 Write Protect Tab
Location Notch
A
Window For Head
Shutter Mechanism
Hub
Index Window
LABEL
1

Fig. 3. External appearance of a 3" floppy disc.

Formats

If we just consider 5¼ inch floppy discs for the moment, there are various types in use, and not all disc drives format discs in the same way. The two formats most commonly used are the 40 track and 80 track systems. As these names suggest, they differ in that one uses 40 tracks on the disc while the other uses 80 tracks, and can consequently store around twice as much information. A 40 track disc has 48 tracks per inch, and therefore uses a band around the disc which is slightly under one inch wide. This band is close to the outside edge of the disc so that each track is as long as possible and optimum reliability is obtained. Although the outer tracks are somewhat longer than the inner ones the same amount of data is usually stored on each track This means that the full storage capacity of the disc is not utilized, but with this system it is much easier to design the disc control system.

With the 80 track format a pitch of 96 tracks per inch is used This gives only about half the track width of the 40 track format which places more stringent requirements on the disc itself as well as the disc drive. Many people seem to imagine that the discs themselves are either 40 or 80 track types, but this is not strictly true. The discs have continuous coating of magnetic oxide over their surface, and the number of magnetic tracks placed onto them depends entirely on the disc drive used. However, there are single density and double density discs available. The difference between the two is the quality of the oxide coating. Single density discs could be used with an 80 track disc drive, but reliability might not be all one would wish and it is advisable not to try this. It would be perfectly alright to use double density discs with a 40 track disc drive, and no problems would result from this. The only drawback of doing this is that you would be paying the extra cost of double density discs when the lower cost single density type would be just as good. Quad density discs and drives are now available, but as yet these have not achieved the same degree of popularity as the standard and double density types.

Double density discs should not be confused with double sided discs. The latter are coated with magnetic oxide on both sides, and the coating on both sides is checked and guaranteed to be up to to standard. When used with a double sided disc drive both sides of the disc can be utilized, giving double the storage capacity of an equivalent single sided type per disc. If you look at a single sided disc you will almost certainly find that it has the head window cut on both sides, and that both surfaces of the disc are coated magnetic oxide. All the 5¼ inch floppy discs currently manufactured are of this type. This is not to say that you can use both sides of the disc by using one side and then turning it over to use the other side. This will not work as the write enable notch and index window are only cut on one side of the protective sleeve. If you turn the disc over they will therefore be on the wrong side of the disc.

Not all 5¼ inch disc drives use either the 40 or 80 track formats, and with home computers in particular it is quite common for the disc format to be one devised by, and unique to, that particular computer manufacturer. However, the CP/M user is unlikely to encounter one of these formats.

A common cause of problems when first using a disc based system is that of not realising the necessity of formating each disc before it can be used. A blank disc is unusable for storing information as the disc drive and operating system rely on the disc providing information to guide the recording/playback head. This information consists basically of recorded numbers to mark out the track positions and the sector boundaries. This is done with the aid of program called a "formatter", and no technical skill is required on the part of the user. You just place the new disc into the disc drive, run the formatter, and when the program has finished running the disc is ready for normal use. CP/M systems are normally supplied with a utility program for formatting blank discs, and this program might also permit copying of discs. A useful feature of the CP/M formatter/copier provided with the Amstrad CPC6128 is its ability to copy discs even if the system has only a single disc drive. It can can also copy a disc, formatting the disc onto which the copy is being made during the copying operation if necessary. However, the formatter and any copying or similar utility programs are provided by the computer manufacturer, and consequently vary in nature according to which particular machine you are using.

Another function of the formatter is to generate the disc directory, although obviously it just lays down the framework of the directory, and it initially contains no entries.

Other Disc Sizes

So far we have not considered standard 8 inch disc drives, but in principle these are no different to the 5¼ inch mini-floppy discs, which are really just scaled down version of the original type. Most 8 inch disc drives use 77 track format and give a storage capacity which is roughly comparable to an 80 track mini-floppy disc system. Unless you obtain secondhand equipment you are not likely to obtain an 8 inch disc system as they are a feature of few (if any) new systems.

You are much more likely to encounter one of the relatively new disc systems based on 3 inch or 3½ inch discs, but again, these are the same in principle as 5¼ inch systems. One slight cause of confusion with some 3 inch types is that they are "reversible" rather than double sided. All that this means is that both sides of the disc can be used, but that the drive has a single head. The second side is used by inserting the disc the other way up, like playing the "B" side of an ordinary record or cassette. Normal density 3 inch discs have a capacity of about 250k per side unformatted, but this is reduced to about 170 to 180k of storage space after the formatting information has been placed onto the disc. 3½ inch types have a somewhat greater capacity at 500k unformatted, and around 350k when formatted.

Hard Discs

Hard discs are something that you are not likely to encounter unless you use a mini-computer or a quite advanced personal computer, although prices are dropping and they are not quite the rarity that they once were. The disc is usually made from aluminium coated with magnetic oxide on which data is stored in the same way as on a floppy disc. However, the disc rotates continuously at a fairly high rate of typically 2400 or 3600 RPM. The aerodynamic heads and precision drive mechanisms allow the heads to move over the discs without ever coming into contact with them and causing damage. Obviously the discs also need to be made to a very high degree of precision if they are not to accidentally come into contact with the read/write heads, and the discs must be rigid. The discs must operate in clean air as dust particles could have disastrous consequences for the disc and (or) heads. Hard discs are therefore normally contained in a rigid air tight container, and are not interchangeable in the same way as floppy discs. This is not the major drawback that it might at first appear to be, as a typical hard disc system would have a massive storage capability of 10M (i.e. 1000k), and some some have capacities many times greater than this. Despite this high capacity, any file on the disc can be accessed almost instantly, and the load/save rate is extremely high.

Although the cost of hard disc systems has fallen to the point where they are not only available for upmarket personal computers, but also for several home computers, these "Winchester" disc systems are still too expensive for the vast majority of small computer users. They are certainly well worthwhile for those who can afford them and make full use of their capabilities, which really means small to medium size businesses, or those who are involved in specialised applications such as CAD and CAM.

Getting Started

Once CP/M is loaded into the computer there should be a line of text at the top of the screen which will state the version of CP/M you have loaded, plus perhaps the name of the computer's manufacturer. There might also be a message detailing the amount of free memory space, and this will say something like "61k TPA". 61k is typical of modern CP/M machines, and is the amount needed to run most popular CP/M applications programs "TPA" stands for "Transient Program Area" incidentally. The number of drives that the system has detected connected and operational may also be displayed (eg "Drive is A" for a single drive system or "2 disc drives" for a two drive system). Note that if any of the disc drives are separate from the main computer and have their own on/off switch, it is normally advisable to switch them on before switching on the computer. They should certainly be switched on before booting CP/M. The exact screen messages obtained depends on the particular version of CP/M and the computer manufacturer concerned, but there should also be a prompt which will probably be something like this:-

```
A>
```

This indicates that you are "in" disc drive A, and any command to (say) load a program will be carried out using drive A. You can switch to drive B by typing in:-

```
B: RETURN
```

The carriage return key might be marked "ENTER" on your particular computer, or you might find that both keys are present (in which case operating either will probably do). In this book "RETURN" will always be used to indicate a carriage return, but you must obviously use whichever key actually provides this function. Once this command has been entered the prompt should change to:-

```
B>
```

However, this will only work if a disc is present in drive B and is obviously only possible in a two drive system. The command:-

```
A:
```

can be used to switch back to drive A, and to restore the original prompt, but there should be a disc in drive A when you type the command. In fact you may find that it is possible to repeatedly change from one drive to the other without having discs in the drives, but it is best not to do this as it risks getting the system somewhat confused, and you might have problems getting either the A or B prompt back. If the computer gets well and truly hung-up it might be that a "cold-start" is the only way of regaining control. This simply entails removing all discs from the drives, switching off the computer, turning it on again, and then booting CP/M again. If the computer has some form of reset key it should be possible to regain control by operating this and then booting CP/M again. Most computers have some form of reset key, but in some cases it is a function that is produced by pressing two or more keys (CONTROL, SHIFT, and ESC simultaneously in the case of the Amstrad CPC6128 for example). With CP/M 2.2 you can try a "warm" start by pressing CONTROL and "C" simultaneously, but this assumes that you have the CP/M program on the disc in drive A (not in drive B even if this happens to be the current drive). A warm start is quicker and easier, but will not often work when the computer has become hung-up. With CP/M Plus CONTROL C can be used to abandon a program or command

and take the computer back to the prompt, but if the computer has crashed it might fail to do this.

If you type meaningless commands into the computer it is unlikely to cause a crash, and it will normally just display whatever you have just typed followed by a question mark ("?") to indicate that the command was not understood. The prompt should also be displayed, indicating that the computer is ready to receive the next instruction. If you make a typing error when entering commands, remember that the delete key can be used to rub out characters to the left of the cursor. If you make a mistake near the beginning of a line but do not notice it until you have almost finished typing the instruction, it might be easier to simply cancel the line and start again rather than delete back to the error. This can be achieved by pressing the CONTROL and "U" keys simultaneously. This will not remove the line from the screen, but will move the cursor to the beginning of the next line, and will leave a hash ("#") sign after the unwanted line to show that it effectively deleted. What is probably a better way of doing things is to press the CONTROL and "X" keys simultaneously, as this does erase the line, and avoids the possible confusion that an erased line left on screen might produce. CONTROL and "H" can be used to backspace and delete one character at a time. This is much the same as using the delete key, but whereas the delete key might become inoperational when running some CP/M utilities, CONTROL and "H" should still function properly.

Returning to the changing from drive A to drive B or vice versa, you can select a specific drive by typing "A:" or "B:" after a command. This only changes the drive for this one command though, and once this has been completed the system returns to whichever drive it was "in" before the command was performed. This will be confirmed by the presence of the "A" or "B" screen prompt, as appropriate.

DIRectory

One of the most simple but useful of CP/M commands is "DIR" (directory), and this lists all the files present on a disc. Try placing the CP/M program disc into the current disc drive and typing:-

```
DIR RETURN
```

into the computer. This should give a full directory of the programs present on the disc, which should include a variety of utility programs as well as CP/M itself (but the latter will not be present under the name "CP/M"). It is obviously very helpful to be able to find out exactly what is on a disc, and with CP/M loaded the DIR command should provide a list of the files on any disc (including any non-CP/M types if your computer has an alternative operating system). Each filename (or line of filenames) will be preceded by "A" or "B" depending on which drive the disc was read from. The filenames are in two parts, and the first part is the filename proper. The second part is called an "extension", and it is used to indicate the type of file concerned. As displayed on the screen there will probably be a large space between the two parts of the filename, but when entering filenames into the computer they are only separated by a fullstop ("."). Thus, if one of the files displayed on the screen is:-

```
DISCKIT3 COM
```

it would be selected by typing:-

```
DISCKIT3.COM RETURN
```

into the computer.

The "COM" extension indicates that the file is a command type "BAS" is used for BASIC source programs and "INT" is used for intermediate BASIC programs. You are most likely to create data files when using CP/M, when storing word processor documents, spreadsheet data, or whatever. There is then no need to provide a specific extension for the files, and you can make up your own (e.g. "TMP" for temporary files, "BAK" for backup files, and so on).

Mostly data files are created from within an application program, and it is then a matter of following the instructions with the program. This usually just involves something like selecting "SAVE" from a list of menu options, and the program will then ask for the filename (if you have not already provided one at some stage), after which it saves the file onto a disc and takes the program back to the menu again.

The DIR instruction can be used to look for a specific file, rather than to display a list of all the files on the disc. If you try typing this command with the system disc in the current drive it should respond with the PIP.COM directory listing to indicate that the specified file has been found.

```
DIR PIP.COM RETURN
```

On the other hand, if you try typing:-

```
DIR TEST.BAS RETURN
```

the computer should return the message:-

```
No File
```

since there will be no file called "TEST.BAS" on the disc.

If you try to list the directory of a blank disc the "No File" message will again be displayed to indicate that the disc has a blank directory.

ERAse

This instruction is largely self explanatory, and it erases a specified file. It will fail to do so if the disc is write- protected, as what the ERA command actually does is to overwrite the filename in the directory with nulls. This leaves the file itself intact on the disc until it is overwritten by a new file. Despite this an erased but non-overwritten file is not retrievable unless a utility program for recovering deleted files is available. Obviously this instruction needs to be used with some care in order to avoid accidentally erasing wanted files.

In order to try out the erase command you can use a word processor or other applications program to create some short files and then try erasing them with the ERA command. For instance, if you wished to erase a file called DUMMY.TXT the instruction:-

```
ERA DUMMY.TXT RETURN
```

would be used. An important point to note is that a space is placed between the command and the filename, and with CP/M commands are always followed by a space. If you were to type the instruction as:-

```
ERADUMMY.TXT RETURN
```

the computer would not understand and would simply respond with:-

```
ERADUMMY.TXT?
```

and do nothing.

In the absence of a suitable applications program with which to create some files it is possible to do so using the ED (EDitor) command This is what is called a "transient" command, and it is not a built-in feature of the main CP/M program. Instead it exists on the CP/M disc as a program ("ED.COM") which is loaded when the ED command is issued. This makes it less accessible than commands such as DIR and ERA since it must be present on the disc you are currently using or it can not be loaded into the computer when it is called. If you wish to experiment with the ED program an easy way of doing this is to make a copy of the entire CP/M program disc. It is standard practice to copy program discs anyway, and to keep the original stored safely somewhere. The copy is then used for everyday loading of the program, and if by some means it should become damaged or simply worn-out, the original can be used to make a new copy. For our present purposes it would be best to make a second copy, and to not have this write protected (since we will be using the disc to save text files, and this will not be possible if the disc is write protected). Your computer will probably be supplied with some form of utility program which facilitates easy disc copying (DISCKIT3 in the case of the Amstrad CPC6128 for example), and the computer's manual should be consulted in order to obtain details of the program to be used and the appropriate procedure.

Assuming that you have a non-write protected copy of the CP/M disc, and CP/M has been booted, try typing this command into the computer:-

```
ED DUMMY.TXT RETURN
```

An important point to note here is that the ED command is called simply by using the letters "ED", rather than by using its full program name ("ED.COM"). When the instruction is issued the computer will load the ED program and will search on the disc for the file "DUMMY.TXT". There is no such file on the disc, and so "DUMMY.TXT" will correctly be assumed to be a new file that you wish to create. The computer should respond with something like:-

```
NEW FILE
:*
```

The "*" is the prompt which shows that the editor is ready to receive an instruction. In this case we wish to place some text in the file, and this is accomplished by issuing the "Insert" command. To do this simply type:-

```
I RETURN
```

This should cause the cursor to go to the beginning of the next line, and the "*" prompt will not be present there. This indicates that the editor is ready to receive text rather than instructions. Unless you are using an early version of CP/M the editor will automatically number each line of text. You will therefore find the cursor preceded by "1:", indicating that you are on the first line. Every carriage return will result in the next line number being automatically displayed on the left hand side of the screen. If a line of text is too long to fit onto one line of the screen display it will continue on the next line, but a new line number will not be issued. The editor program is only a very simple type, and it should not be thought of as a word processor. It is more like an ordinary typewriter in many ways. For example, it does not have a word-wrap facility, and if you are in the middle of a word when you reach the end of a line on the screen, the word will simply be split in two and continued on the next line. Of course, you can put in carriage returns in order to prevent this, and each line of text on the screen will then have its own line number. These line numbers only appear on the screen to aid editing incidentally, and if a file is (say) sent to a printer, the printed copy will not include the numbering.

With CP/M 2.2 and CP/M Plus (but not earlier versions) there are two versions of the insert command. If an upper case “I” is used, then all the text will be in upper case. It may not appear this way when you type it in, but this is the form in which it will be stored on disc. Use a lower case “i” if upper and lower case letters are required.

In order to try out the ED command and to create a file for testing purposes, just type in a few lines of text, which need be no more than some random characters. To finish entering text and return to the command mode simply operate CONTROL and “Z” simultaneously. This should cause the “*” prompt to be returned. The “B” (beginning) command can then be used to take the “character pointer”, or “CP”, back to the beginning of the file. The character pointer is not actually a character which is printed on screen, but is a sort of imaginary or invisible cursor. The “T” (“TYPE”) instruction can then be used to print the complete file on screen. This instruction prints out the specified number of lines following the character pointer, and it should be preceded by the number of lines that are to be printed out. If the whole file is to be printed out it is possible to use a hash (“#”) sign in place of the number of lines in the file. These two commands should therefore cause the entire file to be printed on the screen:-

```
B RETURN
# T RETURN
```

In fact these could be combined in a single instruction, like this:-

```
B#T RETURN
```

If you try the command:-

```
B1T RETURN
```

this should print out just the first line of the file. The hash sign or number must be placed ahead of the “T” and not after it or the command will not work.

To save the file on to disc the “E” command is used, and to implement this you simply type:-

```
E RETURN
```

The file is then saved onto disc, the editor program is terminated, and the normal CP/M prompt is returned to the screen. An alternative is to use the "H" command:-

```
H RETURN
```

Try using this one first to enable a few more lines of text to be added, and then use the "E" command to save the file and return to the command, or "console" mode as it is normally called in CP/M terminology. After using CONTROL Z and issuing the "H" command you will be back in the editor's command mode, and must therefore use the "I" command to insert new text. If your version of CP/M is one which gives automatic line numbering this will start at line 1 again. Any lines of text you add will be placed ahead of your original text. It is not placed after it and neither does it replace it.

If you have a twin disc drive system you can produce files using the editor without having to have the editor program on the disc used to save the files. Assuming you are in drive A, place the CP/M program disc in drive A, and place a formatted but blank disc in drive B (or a disc with some empty workspace anyway). Then type this instruction:-

```
ED B:DUMMY.TXT RETURN
```

Note that the instruction to implement the command on drive B must come after the main command, but before the filename. You can then proceed exactly in the same manner as described above. This method is obviously a more convenient way of using the editor since it is not necessary to have the editor program on the same disc as the files. It is obviously only possible with a system that has two or more disc drives though.

In fact this is not strictly true, and if you have a system which runs under CP/M plus it is possible to use one disc drive to act as both drive A and drive B. This is not quite as good as having two disc drives in that it requires a lot of disc swopping that is unnecessary with a two drive system. Also, you must be careful to always have the right disc in the drive, although if a mistake should be made it is normally possible to recover from it quite easily and the computer is unlikely to crash. It is a feature that is a major plus point for CP/M Plus, and it permits single drive set-ups running under this system to undertake disc operations that are not possible under CP/M 2.2 and earlier CP/M versions with a single drive system.

When using one drive to act as two the commands typed into the computer are exactly the same as when using a genuine two drive system. In order to use "ED.COM" on the CP/M program disc to generate a file called "DUMMY.TXT" on another disc the procedure is much the same as that described above apart from the disc swopping. First enter the command:-

```
ED B:DUMMY.TXT RETURN
```

This should be done with the CP/M program disc in the drive, and the computer will load the "ED.COM" program from the disc. It will then provide a message, probably in the form of a running message across the bottom of the screen, requesting that the disc should be placed in drive B, and that any key should then be pressed. The disc onto which the new file is to be placed is then put into the disc drive in place of the program disc, and any key is operated. This should result in the "NEW FILE" message being displayed, together with the "*" prompt. In addition there will probably be a message at the bottom right hand corner of the screen informing you that you are in drive B. The file is then created and saved, exactly as before. Things can be a little confusing after the "E" command has been executed, as the "A" prompt will be returned, but the "Drive is B:" message may still be displayed. However, the system is in drive A, and when a command is issued you will be requested to put the disc in drive A and press any key. The "Drive is B:" message is then replaced with the "Drive is A" message.

When you have produced the file, by whatever means, and have the "A" prompt again, use the "DIR" command to read the directory. With a two drive system you will presumably have the "A" prompt but will have a disc with the newly generated file in drive B. The command:-

```
DIR B: RETURN
```

should therefore be used, rather than just:-

```
DIR RETURN
```

The directory should include a file called "DUMMY.TXT", as one would expect, but there should also be one called "DUMMY.BAK". This second file is not one that you created directly, but is one that the editor generated for you. What happened was that a file called "DUMMY.TXT" was generated when the "H" command was used, and this file had its name changed to "DUMMY.BAK" when the "E" command was issued and the updated version of "DUMMY.TXT" was created. This is a common practice with programs that create files, and the "BAK" (backup) file is created in case a mistake by the user, a hardware fault, a brief power failure, or something of this nature should result in the main file being accidentally damaged and rendered irretrievable.

In fact when the "ED" command was first issued a file called "DUMMY.$$$" was created on the disc, and this file was replaced with "DUMMY.TXT" when the "H" command was issued. It is for this reason that you will probably find it impossible to successfully issue an "ED" command using the write-protected master CP/M disc. When the "ED" command is entered the computer will try to create a file with the appropriate name on the disc, but will be unable to. With CP/M plus you can actually use an "ED" command with the program on a write procted disc, but the computer will provide a message stating that the disc is write-protected, and giving the option of cancelling the command, retrying it, or ignoring the fact that the disc is protected. These options are selected using the "C", "R", or "I" keys respectively, and it is the "I" option that must be used to get into the editor program. The obvious thing to do then is to change discs so that the newly created file can be saved to disc, but simply changing discs and trying to write to the disc will not work as the computer will have logged-on the first disc, and will detect the change in disc. This will prevent the program from executing properly. Changing discs at the wrong time is a common way of causing the computer to hang-up. A so called "warm" boot by using the CONTROL C combination is normally used when changing discs so that the computer is conditioned to accept the new disc.

When you have established that the two files are present you can display them on the screen using the "TYPE" instruction.

```
TYPE DUMMY.BAK RETURN
```

should result in the first version of the file being displayed, and:-

```
TYPE DUMMY.TXT RETURN
```

should result in the final version being printed on-screen. With long files the initial part of the text might scroll up off the top of the screen before you have a chance to read it. Pressing the CONTROL and "S" keys will halt output to the console (screen) to prevent this from happening. Use CONTROL "S" again to restart output to the console, then CONTROL "S" again to halt it, and so on, as required. Things are a little different with CP/M Plus where CONTROL and "S" are used to halt output to the console, and CONTROL plus "Q" are used to restore it again.

Returning to the erase command, the two files can be erased using the "ERA" command plus the two filenames, as follows:-

```
ERA DUMMY.BAK RETURN
```

and

```
ERA DUMMY.TXT RETURN
```

Using the "DIR" command again will confirm that they have both been delected from the disc. The filename must be given in full, including the extension. The command:-

```
ERA DUMMY RETURN
```

will simply produce the response "No File", and no files will be erased.

The "ERA" instruction can be used to erase more than one file at a time. One way of doing this is to use “*” for the filename and to give the extension in full. For example:-

```
ERA *.BAK
```

will erase all files having “BAK” as their extension. It will not erase a file which has the letters “BAK” in the filename, but not in the extension (e.g. “BAK.TMP” or BAKER.COM would not be erased).

Similarly, the “*” symbol can be used to replace the extension with the filename being specified. Thus “DUMMY.BAK” and DUMMY.TXT” could be erased with the single instruction:-

```
ERA DUMMY.* RETURN
```

With CP/M Plus this will produce the response:-

```
ERA DUMMY.* (Y/N)?
```

Either the “Y” key is operated in order to implement the instruction, or the “N” key is pressed in order to cancel it. In actual fact operating any key other than “Y” will cancel the instruction. The point of all this is to greatly reduce the risk of numerous files being obliterated by the operator making a careless mistake. The “ERA” command is one that needs to be treated with great care anyway, but especially so when it is being used in a form that could eliminate a number of files.

An alternative to “*” is to use one or more question marks (“?”). For example, if you generate files called “DUMMY.TXT” and “DUSTY.TXT”, these would both be erased by the instruction:-

```
ERA DU???.TXT RETURN
```

Again, the computer will give you a chance to change your mind before it deletes the two files, and any others that have “DU” as the first two characters and “TXT” as the extension. In fact this is not quite true, as the number of question marks has some influence on which files are erased, and which ones are not.

"DUMMY.TXT" and "DUSTY.TXT" would both be erased by the command:-

```
ERA DU????.TXT RETURN
```

even though there is one more question mark than there are characters following "DU" in either filename. However, the command:-

```
ERA DU??.TXT RETURN
```

would not erase either of them, as there are too few question marks to cover the number of characters after "DU" in either filename.

Incidentally, "*" and "?" can be used with the directory ("DIR") command as well as with the erase one. In case you were wondering if it is possible to wipe all the files from a disc using the command:-

```
ERA *.* RETURN
```

– it is!

This system of using dummy characters rather than a proper file name is know as using "wild cards", and this is a term which will often be encountered when dealing with disc matters.

REName

The "REN" (rename) command is another very useful one, and it simply renames the specified file. It takes the form:-

```
REN new name=current name RETURN
```

In order to try out this command generate a file called (say) "DUMMY.TXT", and then type this command into the computer:-

```
REN NEWNAME.TXT=DUMMY.TXT RETURN
```

Of course, the disc containing the "DUMMY.TXT" file must be in the current drive when the command is issued. Use the DIR instruction to read the directory and check that the name has been changed. Remember that it is only the directory name that is changed, and the file itself has not been altered. This command does not operate by duplicating the file and giving it the new name, but merely changes the filename in the directory. The old filename therefore disappears from the directory. For this instruction to operate properly the old filename has to be given in full, including the extension. Any extension for the new filename must also be included or it will obviously be omitted from the new filename.

CP/M filenames can have up to eight characters plus the extension. Although it is the convention to use a three character extension, it is acceptable to use only one or two (but not more than three). There are several characters which should not be included in filenames, and these are:-

```
. , ; : ? * = [ ] < >
```

STAT

The "STAT" command is used to display status information, and also to assign devices. Here we will only consider it in the former role.

A simple but very useful application of "STAT" is to display the amount of available space on a disc. To use it in this way it is just a matter of typing the command:-

```
STAT RETURN
```

It then gives the number of bytes available on the disc in the current drive (e.g. "Bytes Remaining On A: 128k"). This is a transient command though, and not a built-in type, and it will only work in this most basic form if the "STAT.COM" program is on the disc being tested. With a two drive system the CP/M program disc can be placed in drive A, and the disc to be tested can be fitted into drive B. This command will then display the amount of free space on the disc in drive B:-

```
STAT B: RETURN
```

If you try changing the disc in drive B and repeating the instruction you will get the same answer as before. In order to get the right answer use CONTROL C when the new disc is in drive B, and then issue the "STAT" instruction. The "STAT" command might provide a response such as "R/W Space 123k", or "R/O Space 123k". "R/W" stands for "read/write", and indicates that it is possible to read and write to the disc. "R/O" stands for "read only", and indicates the the disc is write-protected and can only be read.

The "STAT" command does not exist as such in CP/M Plus, but disc status information can be read using the "SHOW" command. With the "SHOW" instruction it is not necessary to implement CONTROL C between disc changes if you are reading the amount of space available on two or more discs. As "SHOW" is a transient command it is still necessary to have the CP/M program in the current drive before this instruction can be used. However, with a single drive system the "SHOW" command can be used with the one disc drive acting as drives A and B, in the same general manner as for the "ED" command and described earlier.

In this form the SHOW command displays the number of free directory entries:-

```
SHOW B:[DIR] RETURN
```

It is assumed here that the CP/M program disc is in drive A, and the disc to be examined is in drive B. Note that the brackets around "DIR" must be of the square type and not the regular variety. By changing "DIR" for "DRIVE" the command will display a lot of information about the characteristics of drive B (total capacity, size of storage blocks, etc.). Using "USERS" as the word in the brackets displays the number of files on the disc, the number of active files, and the user number (which is a topic which will be covered later). If "LABEL" is used as the word in brackets then label information about the disc, where applicable, is displayed. This is again something which will be described in more detail in a later chapter.

Returning to CP/M 2.2 and the "STAT" instruction, it can be used to give detailed information about a particular file. For example, with the system disc in the current drive try typing this command:-

```
STAT PIP.COM RETURN
```

The computer’s response should be a screen display something along the following lines:-

```
Recs  Bytes    Ext      Acc
58    8k       1        R/W  A:PIP.COM
Bytes remaining on A: 7k
```

The “Recs” entry shows how many 128 byte blocks of the disc (known in CP/M as “records”) have so far been taken up by the file, and the “Bytes” figure shows how many kilobytes (1 kilobyte or 1k=1024 bytes) of disc space are occupied by the file. Disc space is allocated in 1k blocks, even if only one byte of one record is actually needed, and dividing the “Recs” number by 8 will often give a somewhat lower figure than the “Bytes” number. In fact with some disc systems, particularly hard discs, disc space may be allocated in larger blocks of 2k, 4k, or even more. Originally the 128 byte records corresponded to the 128 byte sector size of the disc, but with modern disc formats this relationship does not always exist (although this is of only academic importance to the user).

A byte is an 8 bit binary number, which equates to a decimal (integer) number in the range 0 to 255. When applied to a computer’s memory we are really talking in terms of the number of 8 bit binary numbers that can be stored, and the situation is the same with discs. Although storing numbers in the range 0 to 255 may seem a rather limited and pointless excercise, these numbers can represent program instructions, they can be combined to represent very large numbers equivalent to decimal numbers many digits long, or they can represent text characters. In this last role computers normally use the ASCII (American Standard Codes for Information Interchange), or an ASCII based set of codes. The number in each byte corresponds to a particular character (the number 85 is the code for the letter “U” for instance). With a text file the “Recs” and “Bytes”“ figures give a fairly accurate idea of the number of characters in the file (bearing in mind that things such as spaces and carriage returns count as characters).

The blocks of data on the disc are grouped in 16k chunks called "extents", and the "Exts" field details the number of 16k blocks allocated to a file. Of more interest is the "Acc" field, or "access" field. This can be read/write (R/W) or read-only (R/O), depending on whether or not it is possible to write to a disc, or it can only be read. So far we have only considered write-protected discs in the sense of one that is physically write-protected by a notch (or whatever) in the disc's casing. An individual file can also, be write protected by placing a special code onto the disc. Before writing to a file and altering it, CP/M checks to see if this code is present, and if it is, it aborts the write operation. It is also possible to put a code on a disc that write-protects the whole disc, which is effectively a sofware equivalent of a write-protect tab.

The write-protect code is placed on the disc using the "STAT" command. To try this out, put a copy of the CP/M system disc (on a mechanically non-write protected disc) into the current drive and type:-

```
STAT PIP.COM $R/O RETURN
```

An on-screen message stating something like "PIP.COM Set To R/O" will be placed on the screen. You can then use the "STAT" command to read the status of "PIP.COM" again, and it should have been changed to "R/O". If you now try to alter the "PIP.COM" file in some way, such as by trying to erase it using the "ERA" command, the computer will fail to do so and will respond with an error message such as:-

```
Bdos Err On A: File R/O
```

"Bdos" incidentally, stands for "Basic Disc Operating System".

The "PIP.COM" file can be set back to being a read/write type using the "STAT" instruction again, shown as follows:-

```
STAT PIP.COM $R/W RETURN
```

Using the "STAT" instruction to once again read the status of the "PIP.COM" file should reveal that it has been set back to being a read/write type.

It is permissible to use wild cards with the "STAT" instruction, and the instruction:-

```
STAT *.COM RETURN
```

for instance, would provide details of any file on the disc in the current drive and having "COM" as its extension. Apart from information about the specified file or files, "STAT" also displays the amount of unused storage space on the disc (e.g. Bytes Remaining On A: 7k).

The "STAT" instruction can be used to set a file to "SYS" ("system") status, which makes it undetectable to the "DIR" command. It can also set the status from "SYS" back to "DIR" again, so that it can be listed by the directory. Thus the instruction:-

```
STAT PIP.COM $SYS RETURN
```

would make the file "PIP.COM" a system file. Reading the directory for the disc should result in no sign of "PIP.COM". Information on "PIP.COM" can still be read using the "STAT" instruction though, and the "PIP.COM" filename will appear in brackets to show that it is a system file.

```
STAT PIP.COM $DIR RETURN
```

can be used to set "PIP.COM" back to the directory mode, and the "DIR" command should then list it.

SET

When running CP/M Plus the "SET" command provides some of the functions given by "STAT" in other versions of CP/M. This includes the ability to set a file as read/write or read-only, and to set a file as a system type or directory type. This operates in a very similar manner to the "STAT" command, but there are a couple of differences in points of detail. In order to set a file to the read-only mode, rather than using the dollar sign followed by "R/O", "RO" in square brackets is used. Therefore, to set the file "PIP.COM" as read-only the following command would be used:-

```
SET PIP.COM [RO] RETURN
```

It would be set back the the read/write mode using the same command, but with "RW" inserted in place of "RO". Changes from directory to system and vice versa are handled in the same way. For example:-

```
SET PIP.COM [SYS] RETURN
```

would set "PIP.COM" to the system mode, and:-

```
SET PIP.COM [DIR] RETURN
```

would set it back to the directory mode.

After the SET command has finished executing, it displays the filename and its current settings on the screen (e.g. "A:PIP COM set to system (SYS), Read Write (RW)"). It is possible to set both RO/RW and DIR/SYS with a single command, in this fashion:-

```
SET PIP.COM [RO SYS] RETURN
```

This would set the "PIP.COM" file to read-only and system mode.

So far we have only considered write-protection of files, but a whole disc can be protected in much the same way. This instruction would write-protect drive B:-

```
SET B:[RO] RETURN
```

The computer will respond with an on-screen message such as "Drive B: set to Read Only (RO)". It is important to realise that although it is the accepted practice to talk in terms of the drive having been write-protected, it is in fact the disc that has been protected. In other words, changing the write-protected disc for a read/write type will not result in drive B being unable to write to the disc. Placing the read only disc back in drive B will prevent the drive from carrying out write operations.

The "STAT" and "SET" commands are capable of doing more than just the simple functions described here, and they will be described in greater detail in later chapters.

Printer

CP/M provides a useful facility which enables anything printed on the screen to also be sent to a printer. If your system is equipped with a printer, try pressing CONTROL and "P" simultaneously. Thereafter, anything which is printed on the screen should also be sent to the printer, including the prompt. In order to switch off the printer again simply press CONTROL and "P" again.

This facility is probably of greatest use with the "TYPE" and "ED" commands. By enabling output to the printer and then using "TYPE" to display a file on the screen, a printed copy of the file will also be obtained. By getting into the editor program and then using the "I" instruction, as copy is typed into the computer it will be printed out as well as appearing on-screen. This obviously lacks the facilities of a word processor, or even a good electronic typewriter, but it could be useful for making rough notes or writing short letters. CONTROL – "P" must be used when in the "ED" command mode, rather than when in the "Insert" mode. With a modern version of CP/M any line numbers that are automatically displayed on the screen will also be sent to the printer, since this mode of output simply gives a hard-copy of what is printed on the screen. However, when in the "ED" command mode the automatic line numbering can be switched off by typing:-

```
-V RETURN
```

The automatic line numbering can be restored by typing:-

```
V RETURN
```

when in the "ED" command mode. When the "TYPE" command is used, line numbers in a file are not printed on the screen, and they are therefore not sent to the printer either.

Chapter 2

USING PIP

Most of the important fundamentals of CP/M were covered in Chapter 1, but there was one glaring omission in that no mention of "PIP" was made. "PIP" is a program that exists on the CP/M program disc as "PIP.COM", and it can either be used as a transient command or as a program. "PIP"stands for "Peripheral Interchange Program" and its prime purpose is to copy files from one disc to another. However, it can do much more than this, and the file transfers do not need to be from disc to disc. For example the transfer could be from (say) a disc to an RS232C serial port. It is also possible to do such things as joining files together (concatenation), and format text.

PIP Command

If we take "PIP" as a command first, it can be used to make a copy of a file on one disc drive on to the disc in another drive, but rather like the transient command "ED", this is only possible if the "PIP.COM" program is accessible to one of the disc drives. As a simple test of the "PIP" command, make a non-write-protected copy of the CP/M program disc, and then use the directory command to read the directory. One of the files on the disc should be "ERASE.COM", and in order to make a copy of this to a non-write-protected disc in drive B the following instruction would be used (as explained more fully a little later, some of these initial "PIP" examples will only run properly under CP/M Plus, and slightly different methods are required for earlier versions of CP/M):-

```
PIP B:ERASE.BAK=ERASE.COM RETURN
```

This assumes that the program disc is in drive A, and that this is the current drive. The instruction is fairly self explanatory with "PIP" being followed by the name to be given to the copy of the file. As the copy is to be made on drive B rather than the current drive (A), the filename is preceded by "B:". The equals ("=") sign tells the computer to make a copy of the file having the name that follows. The file to be copied is in the current drive (which is still A), and so no drive needs to be specified here.

If you try this instruction and then list the directory of drive B you should find "ERASE.BAK" is listed in the directory. Of course, the disc in B must have sufficient spare storage capacity to accommodate "PIP.COM", and it is probably best to use a blank but formatted disc when carrying out this familiarisation with the "PIP" command. If you list the directory for drive A you should find that "ERASE.COM" is still present there, as copying a file from one disc to another does not destroy the original.

As will probably be apparent to you, the "PIP" command takes this form:-

```
drive letter:new filename=
drive letter:existing filename
```

Of course, the drive identification letter is only required if the filename that follows is to be read from or created on a drive other than the current one.

When using "PIP" in this simple form it will usually be necessary to transfer the "PIP.COM" program onto the disc from which a file is to be copied, or onto the the disc onto which it is to be copied. Using the CP/M program disc in drive A, and the largely blank disc in drive B, as before, try this command:-

```
PIP B:PIP.COM=PIP.COM RETURN
```

This will make a copy of "PIP.COM" on the disc in drive B, retaining the "PIP.COM" filename for the copy. It is essential not to use a different name as "PIP.COM" is the one that the computer will search for when the "PIP" command is executed. Check the directory of drive B to ensure that the copy has been made successfully.

If you now remove the program disc from drive A, and take the disc from drive B and place it in drive A, by using the following command a copy of the "ERASE.BAK" file, called "ERASE.TWO", should be made:-

```
PIP ERASE.TWO=ERASE.BAK RETURN
```

Again, use the directory to check that the file has been copied. It is a good idea to experiment a little with file copying until you are sure that you fully understand what is involved and can easily produce copies. A point worth noting is that it should be possible to copy any disc files successfully, whether or not they are generated by a program running under CP/M. Of course, this assumes that your computer has the capability of running under an alternative operating system, and that the files concerned have not been copy-protected in some way (much games and other software is doctored to prevent easy copying).

With a computer which runs under CP/M Plus and has only a single disc drive it is possible to use the one drive as both drive A and drive B, in much the same way as for the "ED" command which was briefly described in the previous chapter.

PIP as a Program

When used in the form described above "PIP" is somewhat restricted by the fact the "PIP.COM" program must be present on an accessible disc. Even with a twin disc drive system this can be a little awkward, and ideally it should be possible to take any two discs and transfer a file from one to the other. This is in fact possible if "PIP.COM" is run as a program, and in order to do this it is merely necessary to put the CP/M program disc into the current drive and type this instruction into the computer:-

```
PIP RETURN
```

Do not type:-

```
PIP.COM RETURN
```

The computer should respond with a simple message stating "PIP" and the CP/M version you are running. There should also be a "*" prompt to indicate that the program is ready to receive an instruction. The program is then used in much the same way as the "PIP" transient command, but the word "PIP" is not needed at the beginning of each instruction. As an example, suppose that the file called "DUMMY.FLE" on the disc in drive A is to be copied to drive B and called "NEW.FLE". The instruction used to do this would be:-

```
B:NEW.FLE = DUMMY.FLE RETURN
```

Incidentally, there is an abbreviated form of the copy command which can be used if the copy is to have the same filename as the original. The command then takes the form:-

drive letter (for copy) = drive letter:filename

Thus, in order to copy a file called "DUMMY.FLE" from drive A to drive B the instruction shown below would be utilized.

```
B: = DUMMY.FLE RETURN
```

In order to exit from "PIP" and return to the command mode of the main CP/M program it is merely necessary to press the "RETURN" key. The "PIP" program can be used with one disc drive acting as drive A and drive B if you have system which runs under CP/M Plus, just as the "PIP" transient command can. I have to emphasize that this ability to use one drive as both the A and B type is only a feature of CP/M Plus (CP/M 3), and it is not supported by CP/M 2.2 and earlier implementations.

Multiple Files

One way of copying several files from one disc to another is to copy them one by one using the method described previously. If all the files on a disc are to be copied, your computer system might well have been supplied with a disc copying utility program, and then this could be used to copy all the files. If all but one or two files are required, the easiest solution might be to copy the whole disc and then delete the one or two files that are not required. In some cases there is a third alternative available, and this is to use "PIP" plus the "?" and "*" symbols. This operates in essentially the same way as when using these symbols with the "ERA" command, as described in Chapter 1. For example, if the disc in drive A contained the files "DUMMY.FLE", "DUMMY.TXT", "DUMMY.BAK", and "FILE.TXT", the instruction:-

```
B:=DUMMY.* RETURN
```

would copy "DUMMY.FLE", "DUMMY.TXT", and "DUMMY.BAK" from drive A to drive B. The "*" symbol provides a match with any extension, and so any file with "DUMMY" as the main part of the filename is copied across from drive A to drive B. Accordingly, "FILE.TXT" will not be copied, and if the disc in drive A contained a file called "DUSTY.TXT" this would not be copied either, since the first part of the filename does not perfectly match "DUMMY" However, the instruction:-

```
B: DU??Y.* RETURN
```

would result in "DUSTY.TXT" being copied, together with the three "DUMMY" files.

"PIP" can be used to copy all the files on a disc, but this is not quite the same as copying a whole disc. It will not copy the CP/M program itself, and if used to copy a disc which contains the CP/M program it will copy everything else. The CP/M program is effectively invisible to "PIP" as far as file copying is concerned. CP/M is also invisible to the "DIR" command, and you can not check to see if CP/M is on a disc by examining the directory. However, if CP/M is on a disc then it should be possible to load it from the disc and carry out CP/M commands. The CP/M program is usually stored on a couple of reserved tracks of the disc, which must be formatted correctly so that these tracks are reserved ("System" format rather than "Data" format). The formatting software provided with the system should give the option of "System" or "Data" formatting where appropriate.

In order to copy all the files on a disc you simply use the normal file copying instruction, but with "*.*" as the filename. Thus, in order to copy all the files in the disc in drive A across to drive B this instruction would be used (assuming drive A is the current drive):-

```
B: = *.* RETURN
```

Most CP/M systems are supplied with some form of disc copying utility program, and where a whole disc carrying a great deal of data is to be copied it is generally much quicker and easier to use this program. One possible advantage of using "PIP" is that it will probably put the files onto the disc in a more compact form without any large gaps between files. This might slightly speed things up or effectively enable more files to be placed onto the disc than would otherwise be possible, but it does not really provide any great advantage. Note that any disc copying utility provided with the computer will almost certainly copy the CP/M program if it should be present on the disc being copied.

When copying a disc using "PIP", " VO " can be placed at the end of the command. The "V" causes the computer to verify that the copy is an accurate representation of the original. This obviously avoids the possibility of having what is thought to be an accurate copy when it is actually seriously flawed. Unfortunately, the verification process slows down the copying quite significantly, and it might not be considered worthwhile unless accurate copying is very important for some reason. The "O" is used for files which have an end of file symbol, or where you think there might by an end of file symbol. It is not necessary to use both the "V" and the "O" together, and one or the other can be added to the instruction. Incidentally, copying using "PIP" can be stopped at any time by pressing virtually any key of the keyboard, and a message confirming that the copying has been aborted will then be displayed on the screen.

SYSGEN

The "SYSGEN" transient command is one which was not covered in Chapter 1, and it is relevant here in that it permits the CP/M program to be copied from one disc to another. It is not implemented in CP/M Plus, but is available in CP/M 2.2 and earlier versions. It is run as a CP/M program by typing:-

```
SYSGEN RETURN
```

into the computer with the system disc in the current drive. It is then just a matter of following the on-screen prompts.

As mentioned earlier, the methods of copying files described so far can only be guaranteed to operate properly with CP/M Plus, which is more tolerant of disc changes than are the earlier versions of CP/M. If you load CP/M 2.2, then run "PIP", and then try to change the system disc for one which contains the file that you wish to copy, this disc change will not be tolerated by the system, and probably the only way that the computer will be persuaded to progress any further is to replace the system disc in the current drive. If you have a three drive system this is not a great drawback as you can leave the system disc in place in the current drive (which will presumably be drive A), and then use drives B and C for the copying, with the drive indentification letters being inserted into the copying instructions at the appropriate points. Unfortunately, few users of CP/M (or any other system for that matter) have the luxury of three disc drives.

With a two drive system the simple way around the problem is to always have CP/M and the "PIP.COM" program on one of the discs involved in the copying process. First use the "SYSGEN" command to copy CP/M on to a suitably formatted disc, and then use "PIP" to copy "PIP.COM" onto the disc. The following procedure can then be adopted.

Place the disc containing CP/M and "PIP.COM" in the current drive (which we will assume to be drive A), and put the other disc involved in the copying process in drive B. We will assume for the sake of this example that all the files on A are to be copied across to B. Press CONTROL C to recondition the computer to accept the change to the new discs. Then type:-

```
PIP RETURN
```

to load the "PIP" program. This instruction will then produce the required copying of the files on the disc in A across to the disc in B.

```
B:=*.* RETURN
```

In fact with "PIP.COM" and the CP/M program on one of the discs the copying could be facilitated from the command mode without needing to run "PIP" as a program. This instruction:-

```
PIP B:=*.* RETURN
```

would have the same effect as running "PIP" and then using the program command given earlier.

With a two drive system running under CP/M 2.2 or an earlier version it is possible to copy from a one disc to another when neither of them contain the CP/M program or "PIP.COM", but this can only be achieved by a slightly roundabout method. In fact it is necessary to first copy the file onto a disc which does contain CP/M and "PIP.COM", and to then copy the file from here onto the disc which is its final destination. The unwanted copy on the system disc is then erased. Of course, this method is only possible if the disc used for temporary storage purposes has sufficient vacant storage area for the file, and if it is not write-protected.

In order to try out this method, copy the CP/M program and "PIP.COM" onto a disc, and place this in drive A. Make a short file for test purposes on a blank disc, called (say) "TEST.DTA", and place this in drive B. Use CONTROL "C" to condition the computer to these two discs, and then copy "TEST.DTA" onto the disc in A using this instruction:-

```
PIP A:=B:TEST.DTA RETURN
```

Use "DIR" to check that "TEST.DTA" has been copied across to the disc in A. Then place a blank but formatted disc in drive B, and use CONTROL "C" to condition the computer to the disc change. Now use this command to copy the file to the disc in B:-

```
PIP B:=TEST.DTA RETURN
```

Check the directory of drive B to ensure that the transfer has been accomplished properly, and then use this instruction to delete the unwanted version of the file on the disc in drive A:-

```
ERA TEST.DTA RETURN
```

The file transfer is then complete. If you intend to transfer files using this method it is obviously a good idea to keep disc containing the CP/M program, "PIP.COM", and plenty of free storage area, especially for use as the intermediate disc.

Devices

So far we have only considered the use of "PIP" to transfer files between any two logical devices (within the constraints of those devices). In addition to the disc drives, there are four other logical devices, as listed below (this is for CP/M 2.2 and earlier versions, and not CP/M Plus):-

CON:	The console
PUN:	A punch device
RDR:	A reader device
LST:	A listing device

CP/M has its origins fairly early in computing history, and this is reflected in the "PUN:" and "RDR:" names, which were originally high speed paper punch and reader machines. While it is highly unlikely that you will ever use a computer equipped with either of these, "PUN:" and "RDR:" will still be available in your system, but will be used for other purposes (such as a serial port). "LST:", the listing device, would normally be a printer, and the "CON:" or console device is the keyboard and monitor. "PUN:" and "LST:" are for output only, "RDR:" is for input only, but "CON:""is for input and output.

These are called logical devices so that they can be differentiated from actual, physical devices. There are four physical devices associated with each logical device, and this operates in the manner shown in Figure 4. Each logical device connects to one of four physical devices, with switches being used to select one physical device per logical device. In reality there are no switches as such, and the selection of the physical devices is under software control. On the face of it there can be a maximum of sixteen physical devices in the system, but some of the physical devices can be assigned to more than one logical device, which reduces the true number of options. Also, in a practical system there would not normally be anything like this number anyway. There would typically just be the console and a printer port, plus perhaps a one or two channel serial port of the RS232C type.

Because the hardware present in each system is different, and the method of reading from and writing to various pieces of hardware requires different routines, the CP/M program has to be tailored to suit your particular computer system, so that to

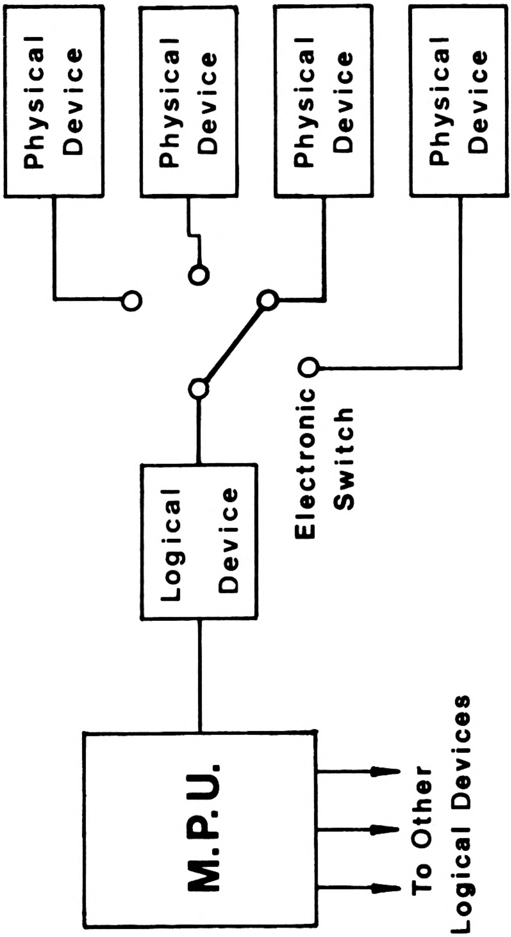

Physical Device
Physical Device
Physical Device
Physical Device
Logical Device
Electronic Switch
M.P.U.
To Other Logical Devices

Fig. 4. Each logical device is actually one of four selectable physical devices

applications programs running under CP/M, the hardware always appears to be the same. This is sometimes referred to as a software interface or a software bus, as the differences between various input/output devices are compensated for in the program without using any additional electronics.

With the CP/M program being tailored to suit the particular hardware present in the computer, it follows from this that changes to the hardware might not be compatible with CP/M. There are many add-on boards available for most popular computers, and if you obtain something like an add-on RS232C serial port which you wish to use with CP/M applications programs, it is essential to make sure that it is compatible with CP/M, or is supplied complete with software which enables it to be integrated into and recognised by the CP/M system. If it is not recognised by CP/M then it is unlikely to be of much practical value, if any.

The physical devices have to be given names so that CP/M can differentiate between them, and a list of the valid physical device names is shown.

BAT:	Input from RDR:, output to LST:
CRT:	Cathode Ray Tube (normally the monitor and keyboard)
LPT:	Line Printer (normally a Centronics type printer output)
PTP:	Paper Tape Puncher (or other output device)
PTR:	Paper Tape Reader (or other input device)
TTY:	Teletype Terminal (or other input/output device)
UL1:	User Defined Listing (output) device
UR1:	User Defined Reader (input) device
UR2:	Second User Defined Reader (input) device
UP1	User Defined Punch (output) device
UP2	Second User Defined Punch (output) device

In a typical set-up “CRT:” (the keyboard and screen) would be assigned to “CON:”, “LPT:” (a Centronics style printer port) would be assigned to “LST:”, and “TTY” (the input and output channels of an RS232C serial port) would be assigned to “RDR:”

and "PUN:". "UR1:" and "UP1:" could be assigned to "RDR:" and "PUN:" respectively to provide an optional second RS232C channel. Some physical devices are likely to be absent, and will be null inputs and outputs. Anything written to these should not cause the system to crash, and reading from a null input simple provides an end-of-file character code. Null inputs and outputs could be useful for testing purposes, as well as offering the possibility of system expansion.

The way devices are handled is slightly different under CP/M Plus and CP/M 2.2. Taking the latter first, this version of the "STAT" command can be used to display the current physical device assignments:-

```
STAT DEV: RETURN
```

With the author's Amstrad CPC6128 computer this results in the following being displayed, and as explained above, these are typical default settings.

```
CON: is CRT:
RDR: is TTY:
PUN: is TTY:
LST: is LPT:
```

However, your system could obviously be different to this.

With CP/M Plus the "DEVICE" instruction is used to display the physical device assignments, as shown below:-

```
DEVICE RETURN
```

The CPC6128 can run both CP/M 2.2 and CP/M Plus, and when running CP/M Plus and trying this instruction the following assignments are displayed:-

```
CONIN: = CRT
CONOUT: = CRT
AUXIN: = Null Device
AUXOUT: = Null Device
LST: = LPT
```

As will probably be apparent from this, CP/M Plus does not use quite the same logical device format as other versions of CP/M. One difference is that "CON:" has been separated into two logical devices, one handling input ("CONIN:" which is the screen) and the other providing output ("CONOUT:" which is the keyboard). "PUN:" and "RDR:" are replaced by "AUXIN:" and "AUXOUT:", which are normally assigned to the input and output of an RS232C serial port. The serial port is an add-on and not a built-in feature of the CPC6128, and consequently these physical devices are null devices. Things are greatly streamlined in CP/M Plus, and physical device names such as "TTY:" and "UP1:" are not used.

With both CP/M 2.2 and CP/M Plus it is possible to reassign physical devices. Again taking CP/M 2.2 first, entering this instruction will give give a brief list of valid "STAT" commands:-

```
STAT VAL: RETURN
```

Remember to have the system disc in the current drive before entering the command, and to use CONTROL "C" first if necessary. The information displayed should include a table showing the legal physical device assignments for each logical device, and it should be something like this:-

```
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:
```

As explained earlier, in most practical CP/M systems the system recognises more physical devices than actually exist, and many of the valid options are not practical options.

The "STAT" command is used to change assignments, and the instruction takes the form:-

```
STAT logical device=physical device RETURN
```

As a simple example, the instruction:-

```
STAT LST:=CRT: RETURN
```

sends output destined for the printer to the screen instead. If you try this and then use CONTROL "P" to switch on output to the printer from the console, you should find that every character thereafter is printed on the screen twice. The first character is the normal screen display, and the second one is the redirected printer output. You have to be careful when reassigning devices as it is easy to hang-up the computer. This will happen if the instruction:-

```
STAT CON:=UC1: RETURN
```

is entered into the computer. This instruction results in the computer looking for its keyboard input from physical device "UC1:", and disconnects the keyboard. The only way out of this might be to remove the discs from the drives, switch the computer off, and then switch it on again and reboot CP/M.

With CP/M Plus the "DEVICE" instruction is used to reassign physical devices in much the same way that "STAT" does with CP/M 2.2. With CP/M Plus there is a much more limited range of theoretical physical devices, but any physical device can be paired with any logical device except where an input/output mismatch makes a pairing nonsensical. This is a very useful scheme of things, making it easy to do something like having the output for a parallel (Centronics) type printer redirected to the serial (RS232C) output if your printer happens to have a serial interface. This example assigns the monitor screen ("CRT") to the parallel printer output ("LST"), but it still leaves the console connected to "CRT".

```
DEVICE LST=CRT RETURN
```

With CP/M Plus there seems to be no need to include the colons after logical or physical device names when using the "DEVICE" command. If you try this example and then enter CONTROL "P", as in our earlier CP/M 2.2 example, every character will be printed on the screen twice (including double line spacing).

If you try to hang-up the computer by cutting off the keyboard using this instruction:-

```
DEVICE CONIN=NULL DEVICE RETURN
```

the system will be too clever for you and will not accept the command. However, this instruction will cut off the monitor screen:-

```
DEVICE CONOUT=NULL DEVICE RETURN
```

This does not hang-up the computer though, and if you type in a command such as:-

```
DIR RETURN
```

the disc drive will be activated, and the computer will carry out the instruction. The directory will be written to nowhere though, and it will not be printed on screen.

Screen Layout

Another function of "DEVICE" is to set the required screen layout. Of course, CP/M 2.2 and earlier CP/M implementations do not have any form of the "DEVICE" instruction, and can not control the screen layout. This instruction will display the current screen layout:-

```
DEVICE CONSOLE [PAGE] RETURN
```

This instruction requires the CP/M program disc to be in the current drive incidentally. It should produce a response something along these lines:-

```
Console width set to 79 columns
Console page set to 24 lines
```

To change the effective page size this instruction is used:-

```
DEVICE CONSOLE [COLUMNS=X LINES=Y]
RETURN
```

where "X" is the required number of columns and "Y" is the required number of lines. If you try out this instruction with quite low numbers for both the columns and lines parameters the effect should be immediately obvious when you type commands into the computer. If you try using the "TYPE" instruction to display a text file on the screen you will notice that the number of columns is controlled by the text file, and not the "DEVICE" instruction. On the other hand, the effective page length will be controlled by the "DEVICE" instruction, with output to the screen being halted every ten lines (or whatever) with an instruction such as:-

```
Press RETURN to Continue
```

being displayed.

Another function of "DEVICE" is to set the operating speed of physical devices, and to enable or disable handshaking. In most cases these are parameters that only apply to a serial interface, and the command:-

```
DEVICE LPT[XON,4800] RETURN
```

would enable handshaking on LPT and set its baud rate at 4800. The instruction:-

```
DEVICE LPT [XOFF,50] RETURN
```

would set the baud rate of LPT at 50 and would disable handshaking. The baud rate is the number of bits transmitted per second with a continuous data stream, and dividing by ten gives the approximate transfer speed in bytes per second. There are a number of standard baud rates available, and these are 50, 75, 110, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 9600, and 19200 baud. Handshaking is where the device receiving data can indicate to the transmitting equipment via a control line that it has received all the data that can currently be handled, and that a hold-off is required. When it is ready to receive data again it indicates this via the control line, and the flow of data is resumed.

With modems it is not normal for handshaking to be implemented, and it can usefully be disabled. With something like a printer which might well be unable to keep up with the flow of data from the computer it is usually necessary to implement handshaking, and it should obviously not then be disabled from software.

In practice, if your computer is equipped with a with serial port, or the manufacturer produces an add-on serial port, the CP/M software may well include a special command to permit easy setting of the baud rate, handshaking, and other factors such as the number of data and stop bits. For example, the Amstrad CPC6128 includes the "SETSIO" command on the CP/M program disc. The manual for your computer should include at least basic details of any command of this type. With such a command available it is obviously preferable to use this rather than the "DEVICE" command. Also, note that lot of possible "DEVICE" instructions are quite meaningless since many peripheral devices use parallel and not serial interfacing. The computer will almost certainly not accept any commands of this type.

You do not really need to understand what is meant by terms such as "stop bits" or "parity", and it is just a matter of ensuring that the word format and baud rate of the computer are set to match the printer or other device connected to the serial port.

Data Transfers

Although so far we have only considered "PIP" for use when transfering data from one disc to another, or between other peripheral devices, it is perfectly possible to transfer data from a disc to a printer or other peripheral device. It is also possible to take data from a peripheral source such as the keyboard and transfer it to disc. This can easily be tested by trying out a few practical examples, and this command will transfer the file "DUMMY.TXT" to the printer.

```
PIP LST:=B:DUMMY.TXT RETURN
```

This assumes that you have produced a suitable test file which is on the disc in drive B, and that the CP/M program disc is in drive A. Of course, you must also have a printer set up and ready to operate as device "LST:". Note that this simply takes the text in the file "DUMMY.TXT" and sends it to the printer, and that the text is not displayed on the screen. Figure 5 helps to illustrate the way in which this operates, and all file transfers operate in the same general manner. First the file is read by the computer and the data is placed in a block of memory, and then it is transferred from here to the printer. Large files are handled in relatively small blocks of data in order to keep within the memory limitations of the computer. This may seem like an unnecessarily complicated way of doing things, but in many cases direct transfer from one device to another would be impractical. With disc drives it is practically impossible to read data one character at a time and transfer it to a printer which can only take characters one by one. It might be possible to devise such a system, but it would probably be very slow and wearing on the disc drives. Using a block of memory as a buffer enables data to be taken from the disc drives (or other peripherals) in conveniently sized blocks, and then transferred to the printer or other device as and when it can accept it.

If you wish to display a file as it is printed, one way to do this is to use the "TYPE" command plus CONTROL P to switch on output to the printer (e.g TYPE B:DUMMY.TXT CONTROL P RETURN). This is not a very good way of doing things in many cases as it will result in the A prompt being printed out at the end of the file (remember that with output to the printer enabled using CONTROL P, everything that appears on screen is normally sent to the printer as well).

What is probably a better way of doing things is to use "STAT" or "DEVICE" to pair the printer with the monitor, so that output for the printer is also sent to the monitor. For instance, with a machine running under CP/M Plus this instruction will give the current device assignments.

```
DEVICE RETURN
```

It also gives the opportuninty to change assignments or to just

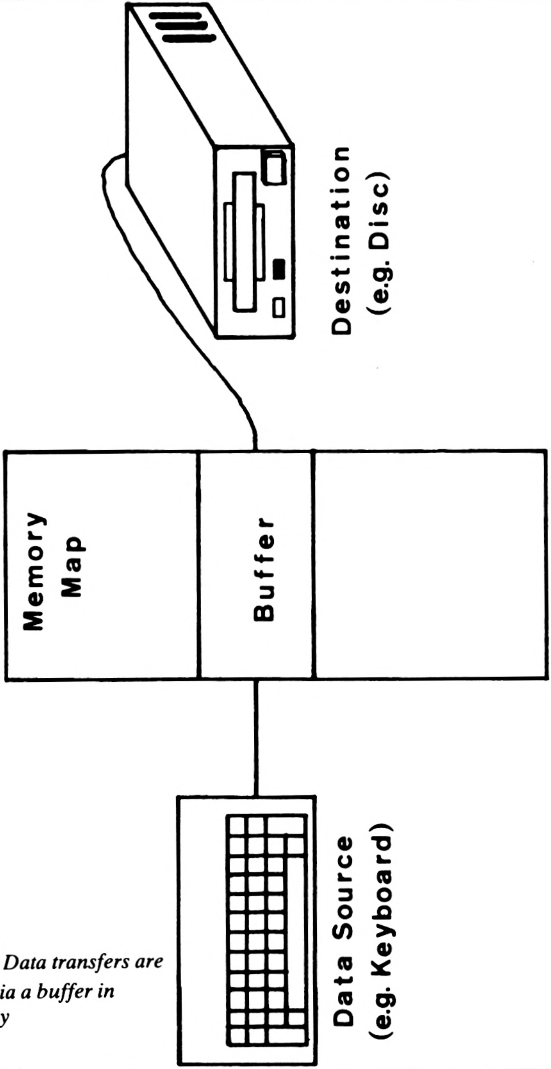

Memory Map
Buffer
Destination (e.g. Disc)
Data Source (e.g. Keyboard)

Fig. 5. Data transfers are made via a buffer in memory

press RETURN in order to leave them unaltered. This instruction would pair the monitor (CRT) with the printer:-

```
LST=CRT,LPT
```

The file dummy text could then be printed on-screen and sent to the printer using exactly the same instruction that was used previously just to give a printed copy.

A couple of "PIP" options were mentioned earlier in this chapter, and there are several others. As one example, "U" can be used to convert all lower case letters to upper case. Try this instruction to print out "DUMMY.TXT" to the printer (use "CON" instead of "LST" if you would prefer the file to be sent to the screen:-

```
PIP LST:=B:DUMMY.TXT U RETURN
```

The "PIP" extensions are described in Appendix 5, and some experimentation with these should soon lead to familiarisation with them.

"PIP" provides an alternative to using "ED" or an applications program to create a text file on disc. The basic way in which this is done is to provide data transfer from the keyboard to a specified disc file (which can be a new one and does not have to be one that already exists). For the sake of this example we will assume that you have a system running under CP/M Plus, with the CP/M program disc in drive A, and a disc with some spare storage capacity in drive B. This instruction will set up the system ready for you to enter text:-

```
PIP B:NEW.TXT=CON:
```

This simply gives the name of the new file ("NEW.TXT") and the drive on which it is to be created (B), and sets "CON:" (the keyboard) as the source. The new file will not be sent to disc character by character as you type it in. As explained earlier, disc drives deal in blocks of data rather than one character at a time. The data typed into the computer is stored in memory until

CONTROL and Z are pressed to indicate that the file is complete. The file is then stored on disc and the computer is returned to the command mode. When entering text you will probably find that RETURN and ENTER do not provide both a line-feed and a carriage return. However, these can be obtained by using CONTROL J and CONTROL M respectively.

When the file has been completed and stored on disc, the "DIR" command can be used to check that the new file has been created, and "TYPE" can be used to print it on the screen. If you try repeating the instruction, adding some more text to "NEW.TXT", and then use "TYPE" to display the file on the screen, you will find that the new text has replaced the original rather than being added to it.

Concatenation

Concatenation is simply the joining together of two or more files to make one large one. This is mostly done with text files, and these are the only type most CP/M users will deal with. Text files have an "end of file" (EOF) code as the final character, and it is this code that is generated when CONTROL "Z" is typed at the end of a file. In order to join two text files end-to-end it is not just a matter of placing the two files on disc as a single entity with no gap in between, but it is also necessary to eliminate the end of file character at the end of the first file. Concatenation of text files is perfectly straightforward with the "PIP" command, and it automatically removes this end of file character.

In order to try out concatenation, generate three short text files called "FILE1.TXT", "FILE2.TXT", and "FILE3,TXT". With the CP/M program disc in drive A, and the file disc in drive B, this instruction will generate a file called "BIG.TXT" by joining "FILE1.TXT" and "FILE2.TXT".

```
PIP B:BIG.TXT=B:FILE1.TXT,B:FILE2.TXT
RETURN
```

The "DIR" instruction can be used to confirm that "BIG.TXT" has been generated on drive B, and then with "TYPE" the new combined file can be displayed on-screen so that a check on its contents can be made.

More than two files can be concatenated simply by including the filenames in the instruction with a comma between each one. This instruction would generate a file called “VERYBIG.TXT” from “FILE1.TXT”, FILE2.TXT”, and “FILE3.TXT”.

```
PIP B:VERYBIG.TXT=
B:FILE1.TXT,B:FILE2.TXT,B:FILE3.TXT
RETURN
```

Again, “DIR” and “TYPE” can be used to check that the new file has been produced correctly. Note that “B:” must precede every filename, as the files being read and the one being created are all on drive B, while it is drive A that is the current one. In fact it might be easier to set drive B as the current drive, and then use this modified version of the instruction.

```
A: PIP VERYBIG.TXT=
FILE1.TXT,FILE2.TXT,FILE3.TXT RETURN
```

It is not necessary to give the concatenated file a new filename, and it is perfectly acceptable to use the name of one of the constituent files if desired. For example, this instruction would concatenate “FILE1.TXT”, “FILE2.TXT”, and “FILE3.TXT” to generate a new “FILE1.TXT” which would overwrite and replace the original file of that name.

```
A: PIP FILE1.TXT=
FILE1.TXT,FILE2.TXT,FILE3.TXT RETURN
```

When using “PIP” to concatenate files you must bear in mind that a new and probably quite large file is being generated, and that the disc must have sufficient storage capacity to accommodate it. If the constituent files are no longer required they can be deleted using the “ERA” command, but both the old and the new files will still be on the disc immediately after concatenation, and it must still have sufficient capacity for all of them. Of course, provided it is within the limitations of your system, the new file can be generated on a different disc to the one from which the constituent files are read.

Partial Copies

In most cases you will probably wish to copy complete files, but it is possible to copy just a portion of a file if this should be necessary for some reason. However, this can only be done in a way that might not be applicable to every situtaton. The basic method is to specify the character string where the copying process must start, and a second string where it must cease. The strings can be any series of characters that occur within the text file, but in practice these would normal be words, or possibly simple codes devised by the user specifically to enable portions of a file to be selected. In order to demonstrate partial copying use "ED" to make the following text file called "PAGES.TXT".

```
Page 1
This is line 1 of the first page.
This is the second line of the first page.
And this is the third line.
Page 1
Page 2
This is line 1 of the second page.
This is the second line of the second page.
And this is the third line.
Page 2
Page 3
This is the first line of the third page.
This is the second line of the third page.
And this is the third line.
Page 3
```

When using "PIP" as a command it will often not work properly when making partial copies as it effectively turns lower case letters in the file to upper case types, preventing a proper match from being obtained. It is consequently much easier to use "PIP" as a program when carrying out this type of copying operation.

With the CP/M program disc in drive A, the disc containing "PAGES.TXT" in drive B, and the "PIP" program running, try this instruction:-

```
CON:=B:PAGES.TXT[SPage 1ZQ Page 1Z] RETURN
```

Note that "Z" here is used to indicate that CONTROL and "Z" are pressed simultaneously (which will be printed on the screen as a vertical arrow preceding the "Z" character). This instruction sends "PAGES.TXT" to the console, or prints it out on the monitor in other words, but the "S" option tells the computer to start at the first occurance of the text string "Page 1" rather than at the beginning of the file. The "Q" option tells the computer to finish at the string "Page 1" rather than at the end of file character. However, it is the next occurance of "Page 1" that causes output to the monitor to be terminated, and the effect of the instruction is therefore to print everything between the "Page 1" at the top of the file, and "Page 1" at the foot of page 1. It is important that the specified text strings exactly match those in the text file (use upper and lower case letters to match the strings in the text file, and do not omit spaces). Another point to watch is that the "Q" string only occurs at the point in the file where you wish output to cease, and that it does not occur earlier. Output will be halted at the first occurance of the string, whether that is the one you had in mind or not.

By entering this instruction it is possible to pick out just Page 3 and to print it on the screen.

```
CON:=B:PAGES.TXT[SPage 3ZQPage 3Z]
RETURN
```

An alternative way of achieving the same thing is to simple print the file from the first occurance of "Page 3" to the end of the file, as in this instruction:-

```
CON:=B:PAGES.TXT[SPage 3Z] RETURN
```

There is no equivalent to this in our earlier example where Page 1 was printed out, and telling the computer to simply print everything as far as "Page 1" would not have the desired effect. It would stop at the first occurance of "Page 1", and not the second. As will probably be obvious to you by now, both the stop and start strings are included in the portion of the file that is printed out.

Simply by using "LST:" instead of "CON:" it is possible to make a disc copy of part of a file, as in this example which will make a copy of Page 2 of "PAGES.TXT", called "PAGE2.TXT".

```
B:PAGE2.TXT=B:PAGES.TXT[SPage 2ZQPage 2Z]
RETURN
```

If you try out this instruction, press RETURN afterwards to return to the CP/M command mode, and then use "DIR" and "TYPE" to check that the new file has been created correctly.

Finally

Probably most CP/M users only ever require "PIP" for simple disc to disc transfers, but it is well worthwhile being aware of its other capabilities. The ability to output files to an RS232C port or to read them from this port is a good example of something that can be very useful on occasions. Much of what "PIP" does can seem to be rather fundamental, but it gives a CP/M system great versatility, and its usefulness is perhaps something that is only fully appreciated by those who have struggled with systems that do not provide the same features.

Chapter 3

SET AND USERS

In the first two Chapters of this book most of the important features of CP/M have been described, but there are still some that have received only superficial treatment, or have not been mentioned at all. In this chapter we will deal with some of these omissions, covering such things as further uses of the "SET" command and user areas.

Passwords

The "SET" command was briefly covered in Chapter 2, but only its functions having "STAT" equivalents were discussed. Another function of "SET" is to assign a password or label to a disc or to a file. Bear in mind that "SET" only exists in CP/M Plus, there is no equivalent to it whatever in earlier versions of CP/M, and the examples provided here will only run if your system operates under CP/M Plus.

In order to use passwords it is first necessary to switch on this facility. We will assume that the CP/M program disc is in drive A, and that a disc with a copy of (say) "PIP.COM" is in drive B. In order to protect "PIP.COM" with a password we must first enable protection on the disc in B. This would be achieved using this instruction:-

```
SET B: [PROTECT=ON] RETURN
```

We must now assign a password to "PIP.COM", and note that the file must exist first and then be given a password, rather than assigning a file to a password. For the sake of this example we will use "KEY" as the password. This would be assigned to "PIP.COM" using this instruction:-

```
SET B:PIP.COM [PASSWORD=KEY] RETURN
```

There are four different levels of protection afforded by the use of a password, and details of each one are as follows:-

READ This gives the highest degree of protection, and the correct password must be given in order to read, copy, write to, delete, or rename the protected file.

WRITE This permits reading and copying of the file, but it is not possible to write to the file. Apart from straightforward writing to the file to modify it, this also prevents deleting and renaming. When using passwords bear in mind that it is not the file itself that is altered in any way, but the directory. The only normal way of reaching a file is via the directory, and a password system at the directory therefore protects the file. Deleting and renaming involve writing to the directory, and are therefore prevented by the "WRITE" mode. Of course, by using software which directly controls the disc drives it is possible to read from or write to any disc, and this form of protection is not unbeatable. However, it does prevent all but the most expert of computer users from reading, copying, or adulterating any protected files.

DELETE This only prevents deleting or renaming of the file without the password. The file can be read or modified without having to give the password.

NONE In this mode no password exists for the file. By setting this mode any existing password can be deleted.

If you try the simple example given above, the computer will provide an on-screen message stating that "PROTECT =READ", and it obviously makes sense to have the default setting the one that gives the highest level of protection. If you use the command:-

```
DIR B: RETURN
```

the "PIP.COM" entry should be listed on the screen. However, if you try to operate on the drive B "PIP.COM" file in any way you will be required to enter the password. To test this out, try erasing "PIP.COM":-

```
ERA B:PIP.COM RETURN
```

When challenged to give the password give the wrong one and press RETURN. This should result in the appropriate error message, and if you list the directory for drive B the "PIP.COM" file should still be intact. Then try repeating the procedure, but give the correct password this time. This time there should be no error message, and when the directory is listed "PIP.COM" should be absent.

Using "PIP", make another copy of "PIP.COM" on the disc in drive B, and then give the file "KEY" as its password again. In order to give a different level of protection the "SET" instruction is used again, as in this example:-

```
SET B:PIP.COM [PROTECT=DELETE] RETURN
```

Of course, this instruction will not be carried out immediately, and in order to change the degree of protection it is necessary to give the password when it is requested. You will notice that the password is not printed on the screen when you type it in, which reduces the risk of someone discovering it. When the password has been entered, press the RETURN key, and the computer should then respond with a message such as:-

```
B:PIP COM Protection=Delete
```

to confirm that the instruction has been carried out.

It is possible to read the "PIP.COM" file with only the delete level of protection, and the command:-

```
TYPE B:PIP.COM RETURN
```

should produce a line of characters on the screen without having to enter the password. However, trying to delete "PIP.COM" with this instruction will only be successful if the correct password is entered when it is requested.

```
ERA B:PIP.COM RETURN
```

For the sake of this exercise, give the wrong password so as to leave the "PIP.COM" file intact on drive B, and then try this instruction:-

```
SET B:PIP.COM[PROTECT=NONE] RETURN
```

Provided you supply the correct password when requested, this removes the pasword protection from "PIP.COM" altogether, and it can then be read, altered, copied, and erased in the normal way. As an example:-

```
ERA B:PIP.COM RETURN
```

would erase it from the directory.

The "SET" command accepts wild cards, and an instruction such as:-

```
SET B:*..*[PROTECT=ON, PASSWORD=KEY,
PROTECT=READ] RETURN
```

would give all the files on the disc in drive B the word KEY as their password, and the "READ" level of protection. Try copying some files from the CP/M program disc across to the disc in drive B, say "PIP.COM", "ERASE.COM", and "RENAME.COM" and then enter the instruction given above. The computer should list the three files, giving "KEY" as the password and "READ" as the degree of protection for all three. Note that in this single instruction three parameters have been set by including a comma and a space between each one.

Although using one password for numerous files may not seem like a very good idea, and it does reduce the degree of security, it does have the advantage of being more manageable. Using a number of different passwords would make it necessary to write them down somewhere unless you have a particularly good memory, and this would compromise security at least as much as having a single password. In the example given above the files all have the same degree of protection, but it is of course, possible to modify the level of security of individual files. This instruction would set "ERASE.COM" to the write level of protection.

```
SET B:ERASE.COM[PROTECT= WRITE]
RETURN
```

An important point to note is that the four levels of security are applied to files, and that they can not be applied to a disc. An instruction of this type will not be accepted by the system:-

```
SET B:[PROTECT=WRITE] RETURN
```

For a disc the only valid instructions are “PROTECT=ON” and “PROTECT=OFF”, and with protection switched on it is possible for each file to be given one of the four levels of protection.

LABELS

The “SET” command can be used to give a disc a label, or name. The SHOW command can be used to display a disc’s label. To try out these versions of the “SET” and “SHOW” commands place a copy of the CP/M program disc in drive A, and a non-write-protected and formatted disc in drive B. In order to give the disc in drive B a label it is merely necessary to use the “SET” command in this way:-

```
SET B:[NAME=LABEL] RETURN
```

If we wish to label the disc in drive B as “DISC1”, then this instruction would be used:-

```
SET B:[NAME=DISC1] RETURN
```

The label can be anything that would be acceptable as a CP/M filename, including an extension if desired.

The label can be protected by a password to prevent any unauthorised person from altering it. The “SET” command assigns the password, and if we wished to have “LOCK” as the password, this instruction would be utilized:-

```
SET B:[PASSWORD=LOCK] RETURN
```

If you now try to alter the name of the disc to (say) "DISC 1234.DTA" using this instruction:-

```
SET B:[LABEL=DISC1234.DTA] RETURN
```

the password will be requested, and the label will only be changed if the right one is provided.

A new password can be assigned to the label using the same instruction format that is used when initially assigning a password. However, the existing password must be given when requested in order to successfully complete the instruction. This form of the "SET" instruction can be used to remove the password:-

```
SET B:[PASSWORD=RETURN
```

Of course the password will be requested and must be typed in before the instruction will delete it.

USER

CP/M Plus supports the built-in command "USER", and this is a fairly simple type. Its basic effect is to divide up the directory of a disc into a maximum of sixteen different user areas which are numbered from 0 to 15. The default setting is 0 (i.e if you do not specify user numbers then 0 is used for files). The general idea is to enable several users to share the computer, with each user having his or her own user number, and his or her own files segregated from those of other users. This is a facility which is probably of more use in a true multi-user system (one where a single computer is shared via a number of consoles), rather than where one computer and console is used by several people. There is actually a multi-user version of CP/M called MP/M, but it has not achieved the same widespread acceptance as CP/M.

If you would like to experiment with the USER command to familiarise yourself with it, place the CP/M disc in drive A and a blank formatted disc in drive B. Then use "PIP" to copy a file from the CP/M program disc to the disc in drive B ("ERASE.COM" for instance). Then type this instruction into the computer:-

```
USER 7 RETURN
```

This should result in the normal screen prompt being modified by the addition of a figure seven ahead of the letter A (i.e, "7A "). If you try reading the directory for drive B it should return a "No File" message, as the "ERASE.COM" file with its user number of 0 is invisible to user number 7. If you enter this instruction:-

```
USER 0 RETURN
```

and then read the directory for drive B again, the "PIP.COM" file should be listed. Switch to any other user number and it will seemingly disappear from the directory again.

One drawback of user numbers is that they can make programs, including CP/M transient commands, invisible to the computer, making it necessary to have a copy of each one for each user. This can be overcome by having files in user area 0 with the "SYS" attribute set. For example, type these three instructions into the computer:-

```
USER 0 RETURN
SET B:ERASE.COM[SYS] RETURN
USER 7 RETURN
```

This sets the "SYS" attribute of the "ERASE.COM" file on the disc in drive B, and the file is in user area 0 of course. The computer is then set to user 7. If the directory for drive B is listed, the "ERASE.COM" file will not be listed as the "SYS" attribute being set renders it invisable to the "DIR" command. It is accessible to any user number though, and the command:-

```
TYPE B:ERASE.COM RETURN
```

should result in the "ERASE.COM" file being displayed on the screen (it will be mostly just random characters interspersed with some recognisable text).

ASM

This is a command which is only likely to be of interest to advanced CP/M users, and it is a form of assembler program. The microprocessor in the computer, which is an 8080 or Z80 type for a CP/M machine, responds to instructions which are in the form of numbers. For an 8 bit microprocessor these are integers in the range 0 to 255, but they are fed to the microprocessor in the form of 8 bit binary numbers. Instructions in this basic form are called "machine code", and programming in true machine code is quite a difficult business. Assembly language is somewhat easier, and this uses mnemonics instead of the instruction numbers. For instance, the mnemonic for the return from subroutine instruction is "RET", which is easier to remember than the instruction number which is 201. The assembler converts the mnemonics into the appropriate instruction numbers, which can be loaded into the computer and run as a machine code program. Programming in assembly language is more difficult than programming in a high level language such as BASIC, and it requires a reasonable knowledge of how the microprocessor operates, together with some understanding of how the system as a whole functions. The book BP 152 "An Introduction To Z80 Machine Code", by the same author and publisher as this book, provides more information on this topic for those who would like to persue the subject further.

For the assembler to function the assembly language program must be placed in a text file having "ASM" as its extension (it will not operate if the wrong extension or no extension at all is present on the filename). The assembler takes the source file and generates two new files from it, one of which is the assembled machine code. It is in a form of numbering known as "hexadecimal", or just "hex", and accordingly the new file is given the same name as the source file. but with "HEX" rather than "ASM" as the extension. The second file is a list of the assembly language program plus the assembled machine code, together with any error messages where appropriate. The purpose of this file is to provide a printable file which is useful when writing and debugging programs, and it is given the original filename plus "PRN" as the extension.

In its most basic form the "ASM" instruction takes this form:-

```
ASM filename RETURN
```

This assumes that the "ASM.COM" program and the file to be assembled are both on the current drive, and that the two new files are to be sent to this drive. In most cases this will not be so, and a fullstop plus three letters are added after the filename (the "ASM" extension is not used in the filename, incidentally). The first letter indicates which drive contains the source file that is to be assembled, while the second letter designates which drive will receive the assembled hexadecimal file. Using the letter "Z" prevents the hexadecimal file from being created. The third letter designates the drive which should receive the "PRN" file, and using a "Z" here again suppresses the creation of the file. Thus it is possible to generate only the "HEX" file or the "PRN" file if desired. Of course, if the "ASM.COM" program is not on the current drive the instruction must be preceded by the letter of the drive where the program is to be found, together with the usual colon.

As a simple example of the "ASM" command, this instruction would assemble the program held in the text file called "LOOP.ASM" on drive B, placing the assembled program in a file called "LOOP.HEX" on drive B. Generation of the "LOOP.PRN" file would be suppressed.

```
ASM LOOP.BBZ RETURN
```

The transient command "LOAD" is used to generate a command file from the assembled "HEX" file. In other words it generates a file with a "COM" extension which can be used as a normal transient command. There is no "LOAD" instruction in CP/M Plus, but an equivalent command is available in the form of "HEXCOM".

"ASM" should not be confused with the "DUMP" command, which gives a hexadecimal listing of the contents of the specified file. For example, this instruction would give a hexadecimal listing of the characters in the file "PAGES.TXT" (with the transient command program "DUMP.COM" accessible on the current drive, and "PAGES.TXT" available at drive B):-

```
DUMP B:PAGES.TXT RETURN
```

With "ASM" the hexadecimal numbers produced are the assembled machine code for the assembly language program in the text file. "LOAD" is a much more simple command, and it simply takes each character in the file, and gives the (hexadecimal) number for its ASCII code. Remember that the text characters are stored by computers as code numbers from 0 to 255 in the ordinary decimal numbering system, or 00 to FF in hexadecimal. The letter "A" for instance, is stored as ASCII code number 65 (decimal) or 41 (hexadecimal). The numbers produced by the "DUMP" command are displayed on a sixteen column format, with addresses down the left hand side of the screen indicating the positions of the characters within the file. These character addresses are in hexadecimal. With CP/M Plus the text characters themselves are displayed at the right hand side of the screen.

The "DDT.COM" program is as an aid to debugging programs, and its use goes well beyond the scope of this publication. It is not included with CP/M Plus incidentally.

"MOVCPM" is a transient command that can be used to produce CP/M with a different TPA size. It is not a command that many CP/M users are likely to require, and it is not included in CP/M Plus.

The "ED" command was discussed briefly in Chapter 1, and two of the appendices give brief details of "ED" commands and control codes. Many CP/M systems are supplied complete with sophisticated word processor programs, and most users soon buy one if suitable software was not supplied with the system. This renders "ED" superfluous to the vast majority of CP/M users these days, and it is probably not worthwhile spending a great deal of time getting to grips with its commands. Most word processors provide the same facilities as a simple editor program, plus a great deal more besides, and are very much easier to use.

GET

This is another transient command which is only available to CP/M Plus users, and which is not included in earlier implementations. Its basic function is to take the console input from a file rather than direct from the keyboard. This is not just a matter of printing the contents of the file on the screen such as when using the "TYPE" instruction, and neither is it the same as calling up a *.COM file (where the contents of the file have to be an executable program rather than text). When "GET" is used, the computer will respond to words in the file just as if they were commands typed in from the keyboard. As a simple example to try out "GET", use "ED" or some other means to produce a text file called "CONTROL.PRG" which simply reads:-

```
DIR RETURN
```

With the system disc in the current drive (A) and the "CONTROL.PRG" file on the disc in drive B, try typing this command into the computer:-

```
GET FILE B:CONTROL.PRG[SYSTEM] RETURN
```

The computer should respond with a message which reads something like "Getting console input from file:B:CONTROL. PRG". It should then print on the screen the instruction read from the file ("A DIR"), and then in response to this it should list the directory for drive A on the screen.

When using "GET", the computer continues to take its input from the file until the end of the file is reached, or until "GET CONSOLE" is read from the file. If you make this modified version of "CONTROL.PRG" and then enter the "GET" instruction used previously, the computer will list the directory for drive A as before, but at "GET CONSOLE" it reverts to the keyboard for input, and the directory for drive B is not displayed.

```
DIR RETURN
GET CONSOLE RETURN
DIR B: RETURN
```

When input is no longer taken from the file the computer gives a "stopped" message, plus a message which states something like "Getting console input from console". If you try making a version of "CONTROL.PRG" with the "GET CONSOLE" line removed, on running the “GET” instruction again you should find that both the A and B directories are listed on the screen.

This form of the “GET” command switches off the “echo”, so that instructions read from the file are not printed on the screen (but the screen prompt will still be printed).

```
GET FILE B:CONTROL.PRG [NO ECHO SYSTEM]
RETURN
```

The word “ECHO” is used to reinstate this facility.

An alternative way of using the “GET” command is to first just enter:-

```
GET RETURN
```

and then at the prompt enter the filename plus any required options (NO ECHO and (or) SYSTEM). So far we have only considered “GET” when used in conjunction with the “SYSTEM” option, which causes it to immediately go to the specified file and execute the instruction or instructions it contains. Without the “SYSTEM” option set, the “GET” command works in a more indirect way. This example should help to clarify matters. Make up a file called “CONTROL.PRG” which reads as follows:-

```
PIP.COM RETURN
```

and place this in drive B. With the CP/M program in the current drive (A), enter this command:-

```
GET FILE B:CONTROL.PRG RETURN
```

The computer will respond with a message which reads something along the lines of “Getting console input from file: B:CONTROL. PRG”, but the “A” prompt will return and the computer will appear to have done nothing apart from accessing the disc drives. However, if you then enter the command:-

```
TYPE RETURN
```

the computer will print the contents of "PIP.COM" onto the screen (this will be a line of largely meaningless characters). Normally if "TYPE" is entered without specifying a filename, the computer will request one. In this case it does not though, but instead takes the necessary information from "CONTROL.PRG", and prints the contents of "PIP.COM" on the screen. The effect of the "SYSTEM" option is to make "GET" operate on the whole system, rather than just on a program.

PUT

By using the "PUT" command it is possible to send output for the printer or the console to a specified file. The output can be echoed to the console or printer as well if desired. This is a command which is only available with systems that operate under CP/M Plus incidentally.

As an initial experiment with the "PUT" command, place the CP/M program disc in the current drive (which we will assume to be drive A), and put a formatted disc with some spare storage capacity in drive B. Then enter this instruction:-

```
PUT CONSOLE OUTPUT TO FILE B:SEND.TXT
RETURN
```

If a file called called "SEND.TXT" should already exist on drive B, the computer will inform you of this fact and ask whether or not to delete it (so that it can generate an initial file called "SEND.$$$" in its place). We will assume here that no file of that name already exists, or that if it does the computer is told to delete it (by pressing the "Y" key) so that the instruction can go ahead. If you then enter this instruction into the computer:-

```
TYPE PIP.COM RETURN
```

the computer should type on the screen the contents of "PIP.COM", and it should also send this information to the file "SEND.$$$" and change its name to "SEND.TXT". You can check this by getting the computer to display the contents of "SEND.TXT" using this instruction:-

```
TYPE B:SEND.TXT RETURN
```

If you repeat this but use NO ECHO as the option at the end of the "PUT" instruction, it will operate in the same manner as before, but when writing the contents of "PIP.COM" to the file it will not also echo it to the console. When sending data for the printer to a file the default setting is for no echo, and if data must be sent to both the printer and the file the echo facility must be enabled using the ECHO option on the "PUT" instruction. Further options are FILTER and NO FILTER (the former being the default setting). Filtering has the effect of translating control characters to printable characters. If you try the examples given above you will probably notice that when "PIPCOM" is echoed to the screen it has fewer characters than when the version of "PIP.COM" stored in "SEND.PRG" is printed on the screen. This is due to the filtering converting unprintable characters (which are actually machine code instructions in the "PIP.COM" program) into control characters. Trying the same instructions but with the NO FILTER option added to the "PUT" instruction, the echoed version of "PIP.COM" should exactly match what appears on-screen when "SEND.PRG" is printed using the "TYPE" command. The SYSTEM option is available with the "PUT" command, and it sends system as well as program output for the console/printer to the specfied file.

SUBMIT

Last, but by no means least, the "SUBMIT" command enables commands in a file with a "SUB" extension to be carried out on execution of the "SUBMIT" instruction. As a simple demonstration, use "ED" or some other means to create a text file called "TRYOUT.SUB" and having these two lines as it contents:-

```
DIR RETURN
DIR B: RETURN
```

With the disc containing "TRYOUT.SUB" in drive B, and a non-write protected copy of the CP/M system disc in drive A, try this instruction:-

```
SUBMIT B:TRYOUT RETURN
```

This should result in the two instructions in "TRYOUT.SUB" being executed, with the directories for drives A and B being displayed on the screen. Note that the "SUB" extension does not need to be included in the "SUBMIT" instruction, and that the file being called must have "SUB" as its extension or the instruction will not be carried out.

The "SUBMIT" instruction can go one stage further than this, with parameters being omitted from the file, and being included in the "SUBMIT" instruction instead. The parameters are called $1, $2, etc. through to $9, and these are analogous to variables in a BASIC program. As a simple example of this facility, make a new version of "TRYOUT.SUB" having these two lines:-

```
DIR $1 RETURN
DIR $2 RETURN
```

Then try this version of the "SUBMIT" instruction:-

```
SUBMIT B:TRYOUT A: B: RETURN
```

This should again list the directories for drives A and B. However, if you try this "SUBMIT" instruction, "TRYOUT.SUB" will then list the directory for drive A twice:-

```
SUBMIT TRYOUT A: A: RETURN
```

The point of all this is to make it easy to carry out long sequences of commands that are frequently used. When executing, the "SUBMIT" command produces a temporary file on the same disc as the one which contains "SUBMIT.COM" (the "SUBMIT" transient command program), and for this reason the disc containing "SUBMIT.COM" must not be write protected. If you try out the last example given above, you should find the temporary file listed in the first directory listing, but not in the second one as it is deleted once the final instruction is reached, and prior to it being carried out. Of course, the disc which contains "SUBMIT.COM" must have some space to accommodate this temporary file.

With computers running under CP/M Plus there is further refinement available when using "SUBMIT". When the CP/M program is booted, it checks for a file called "PROFILE.SUB" on the system disc, and if such a file exists it carries out the instructions it contains. If you have computer which runs under CP/M Plus you can demonstrate this facility by placing a text file called "PROFILE.SUB" on a non-write protected copy of the system disc, and including a few simple instructions on it ("DIR", "DIR B:", etc.). On booting CP/M from this disc, the usual initial screen display should be produced, and the instructions in "PROFILE.SUB" should then be executed immediately.

Finally

Although not all aspects of CP/M have been given in-depth coverage, if you are able to follow the points covered in this book you should be well able to use a CP/M computer system and deal with any minor difficulties that arise. The appendices should be useful as they provide an easy way of checking what control code is needed for a particular function, whether or not a particular command is supported by the version of CP/M you are using, etc.

Appendix 1

CONSOLE AND ED COMMAND CODES

CONTROL C	Warm boot (CP/M 1.4/2.2), Abandon (CP/M 3.0)
CONTROL E	Carriage return (without executing line)
CONTROL H	Backspace and delete
CONTROL J	Line feed
CONTROL M	Carriage return (no line feed)
CONTROL P	Echo screen output to printer (use control P again to switch off output to printer).
CONTROL Q	Resume screen output (CP/M 3.0)
CONTROL R	Retype line
CONTROL S	Halt output to screen (with CP/M 1.4/2.2 use Control S again to restart output)
CONTROL U	Discard line
CONTROL X	Delete from start of line to cursor
CONTROL Z	End of file character

Note that CONTROL H, J, M, R and X are not supported by CP/M 1.4.

Appendix 2

MAIN ED COMMANDS

A	Append a line from the file to the edit buffer memory block (or the number of lines specified in the figure preceding the letter "A")
B	Moves character pointer to beginning of file
–B	Moves character pointer to end of file
D	Deletes character at the character pointer (positive or negative number ahead of the letter "D" results in the specified number of character forwards or backwards being deleted)
E	Ends session and saves file
Fstring	Finds the specified string which must be after the current position of the character pointer, and places character pointer at final character of the string)
H	Performs the E command after which ED is executed again
I	Insert new lines of text after character pointer (with CP/M 2.2 and CP/M 3.0 an upper case "I" gives all upper case characters, and a lower case "i" gives upper and lower case characters)
xK	Delete (Kill) the specified (x) number of lines (use positive number to delete after character pointer, or negative number to delete lines ahead of it)
xP	Print "x" pages of text (with "x" omitted only the current page is printed, positive and negative numbers print pages after or before the character pointer respectively)
Q	Quit from ED (NO *.BAK file is created)
Soriginal	textCONTROL Znew text Searches after character pointer in buffer for original text, and replace it with new text (a number at the end of the command will cause it to be repeated the specified number of times)

xT	Type (display) the characters from the character pointer to the end of the file (where "x" is included, the specified number of lines before or following the character pointer will be displayed depending on whether "x" is negative or positive)
U	All entered characters are translated to upper case (use –U to disable the translation process)
V	Switch on line numbering display (default mode of CP/M 2.2 and CP/M Plus)

Appendix 3

PIP PARAMETER OPTIONS

B	BLOCK MODE TRANSFER. Data is transferred to buffer until an OFF (CONTROL S) character is received. It is then transferred to disc.
Dx	DELETE. Characters after column "x" are deleted during transfer, and the effect is therefore to truncate lines to the specified line length.
E	ECHO. The transfers data is echoed to the console (monitor screen).
F	FORM FEED STRIP. During transfer all form feeds are removed from the file.
Gx	GET FILE FROM USER AREA "X". Self explanatory.
H	HEXADECIMAL. Used for transfering hexadecimal (i.e. ".HEX") files.
L	LOWER CASE. Converts all upper case letters to lower case during transfer.
N	LINE NUMBERS. Adds line numbers to each line that is transferred. Use "N2" to prevent leading zeros being suppressed and to insert a tab space after each line number.
O	OBJECT FILE TRANSFER. When this option is used the end of file character is ignored. This is used when copying a file which is not a text type and does not have a "COM" extension.
Px	PAGE EJECTS. Simply inserts a form feed character every "x" lines. Default value is 60 lines.
Qstring CONTROL Z	QUIT COPYING. The transfer stops when the specified string is reached. CONTROL Z is used to terminate the string.
R	READ SYSTEM FILES. Read files with system ($SYS) status.

Sstring CONTROL Z	START COPYING. Starts copying from the specified string, rather than at the beginning of the file. Use CONTROL Z as the terminator at the end of the string.
Tx	TAB. During transfer the TAB characters are expanded to every "x" columns. CONTROL I generates a TAB character (which is normally fixed at a length of seven characters).
U	UPPER CASE. During transfer all lower case letters are converted to upper case.
V	VERIFY. Verifies that data has been copied correctly, and is only usable when data is being transferred to disc.
W	WRITE. Overwrites read-only (RO) files. Implemented in CP/M 2.2 and CP/M 3.0 only.
Z	ZERO PARITY BIT. Zeros the parity bit on input of each ASCII character.

Appendix 4

CP/M COMMANDS

Command	CP/M 1.4/2.2	CP/M 3.0
ASM	Yes	Yes
DEVICE	No	Yes
DDT	Yes	No
DIR	Yes	Yes
DUMP	Yes	Yes
ED	Yes	Yes
ERASE	Yes	Yes
GET	No	Yes
HEXCOM	No	Yes
LOAD	Yes	No
MOVCPM	Yes	No
PIP	Yes	Yes
PUT	No	Yes
REN	Yes	Yes
SAVE	Yes	Yes
SET	No	Yes
SHOW	No	Yes
STAT	Yes	No
SUBMIT	Yes	Yes
SYSGEN	Yes	No
TYPE	Yes	Yes
USER	No	Yes

Appendix 5

EXTENSION TYPES

ASM	Assembly language program in text file form
BAK	ED backup file (also used by other applications programs)
BAS	BASIC program source files
COM	Transient command
HEX	Hexadecimal (machine code) program file ready for LOAD command
INT	Intermediate BASIC program file
PRN	Assembly language program listing
SUB	Text file with built-in CP/M or transient commands
$$$	Temporary files created and overwritten by ED and other and other applications programs

Notes

Notes

Please note following is a list of other titles that are available in our range of Radio, Electronics and Computer books.

These should be available from all good Booksellers, Radio Component Dealers and Mail Order Companies.

However, should you experience difficulty in obtaining any title in your area, then please write directly to the publisher enclosing payment to cover the cost of the book plus adequate postage.

If you would like a complete catalogue of our entire range of Radio, Electronics and Computer books then please send a Stamped Addressed Envelope to:

BERNARD BABANI (publishing) LTD
THE GRAMPIANS
SHEPHERDS BUSH ROAD
LONDON W6 7NF
ENGLAND

160	Coil Design and Construction Manual	£2.50
202	Handbook of Integrated Circuits (ICs) Equivalents and Substitutes	£2.95
205	Hi-Fi Loudspeaker Enclosures	£2.95
208	Practical Stereo and Quadrophony Handbook	£0.75
214	Audio Enthusiast's Handbook	£0.85
219	Solid State Novelty Projects	£0.85
220	Build Your Own Solid State Hi-Fi and Audio Accessories	£0.85
221	28 Tested Transistor Projects	£2.95
222	Solid State Short Wave Receivers for Beginners	£1.95
223	50 Projects Using IC CA3130	£1.25
224	50 CMOS IC Projects	£2.95
225	A Practical Introduction to Digital ICs	£1.75
226	How to Build Advanced Short Wave Receivers	£2.95
227	Beginners Guide to Building Electronic Projects	£1.95
228	Essential Theory for the Electronics Hobbyist	£2.50
BP1 + 14	First & Second Books of Transistor Equivalents & Substitutes	£3.50
BP2	Handbook of Radio, TV, Industrial and Transmitting Tube and Valve Equivalents	£0.60
BP6	Engineer's and Machinist's Reference Tables	£1.25
BP7	Radio and Electronic Colour Codes Data Chart	£0.95
BP27	Chart of Radio, Electronic, Semiconductor and Logic Symbols	£0.95
BP28	Resistor Selection Handbook	£0.60
BP29	Major Solid State Audio Hi-Fi Construction Projects	£0.85
BP33	Electronic Calculator Users Handbook	£1.50
BP34	Practical Repair and Renovation of Colour TVs	£2.95
BP36	50 Circuits Using Germanium Silicon and Zener Diodes	£1.50
BP37	50 Projects Using Relays, SCRs and TRIACs	£1.95
BP39	50 (FET) Field Effect Transistor Projects	£1.75
BP42	50 Simple LED Circuits	£1.95
BP44	IC 555 Projects	£2.50
BP45	Projects in Opto-electronics	£1.95
BP48	Electronic Projects for Beginners	£1.95
BP49	Popular Electronic Projects	£2.50
BP53	Practical Electronics Calculations and Formulae	£2.95
BP54	Your Electronic Calculator and Your Money	£1.35
BP56	Electronic Security Devices	£2.50
BP58	50 Circuits Using 7400 Series ICs	£2.50
BP59	Second Book of CMOS IC Projects	£1.95
BP60	Practical Construction of Pre-amps, Tone Controls, Filters and Attenuators	£1.95
BP61	Beginners Guide to Digital Techniques	£1.95
BP62	The Simple Electronic Circuit & Components (Elements of Electronics – Book 1)	£3.50
BP63	Alternating Current Theory (Elements of Electronics – Book 2)	£3.50
BP64	Semiconductor Technology (Elements of Electronics – Book 3)	£3.50
BP65	Single IC Projects	£1.50
BP66	Beginners Guide to Microprocessors and Computing	£1.95
BP67	Counter Driver and Numeral Display Projects	£2.95
BP68	Choosing and Using Your Hi-Fi	£1.65
BP69	Electronic Games	£1.75
BP70	Transistor Radio Fault-finding Chart	£0.95
BP71	Electronic Household Projects	£1.75
BP72	A Microprocessor Primer	£1.75
BP73	Remote Control Projects	£2.50
BP74	Electronic Music Projects	£2.50
BP75	Electronic Test Equipment Construction	£1.75
BP76	Power Supply Projects	£2.50
BP77	Microprocessing Systems and Circuits (Elements of Electronics – Book 4)	£2.95
BP78	Practical Computer Experiments	£1.75
BP79	Radio Control for Beginners	£1.75
BP80	Popular Electronic Circuits – Book 1	£2.95
BP82	Electronic Projects Using Solar Cells	£1.95
BP83	VMOS Projects	£1.95
BP84	Digital IC Projects	£1.95
BP85	International Transistor Equivalents Guide	£3.50
BP86	An Introduction to BASIC Programming Techniques	£1.95
BP87	50 Simple LED Circuits – Book 2	£1.35
BP88	How to Use Op-Amps	£2.95
BP89	Communication (Elements of Electronics – Book 5)	£2.95
BP90	Audio Projects	£1.95
BP91	An Introduction to Radio DXing	£1.95
BP92	Electronics Simplified – Crystal Set Construction	£1.75
BP93	Electronic Timer Projects	£1.95
BP94	Electronic Projects for Cars and Boats	£1.95
BP95	Model Railway Projects	£1.95
BP97	IC Projects for Beginners	£1.95
BP98	Popular Electronic Circuits – Book 2	£2.25
BP99	Mini-matrix Board Projects	£1.95
BP101	How to Identify Unmarked ICs	£0.95
BP103	Multi-circuit Board Projects	£1.95
BP104	Electronic Science Projects	£2.25

BP105	Aerial Projects	£1.95
BP106	Modern Op-amp Projects	£1.95
BP107	30 Solderless Breadboard Projects – Book 1	£2.25
BP108	International Diode Equivalents Guide	£2.25
BP109	The Art of Programming the 1K ZX81	£1.95
BP110	How to Get Your Electronic Projects Working	£1.95
BP111	Audio (Elements of Electronics – Book 6)	£3.50
BP112	A Z-80 Workshop Manual	£3.50
BP113	30 Solderless Breadboard Projects – Book 2	£2.25
BP114	The Art of Programming the 16K ZX81	£2.50
BP115	The Pre-computer Book	£1.95
BP117	Practical Electronic Building Blocks – Book 1	£1.95
BP118	Practical Electronic Building Blocks – Book 2	£1.95
BP119	The Art of Programming the ZX Spectrum	£2.50
BP120	Audio Amplifier Fault-finding Chart	£0.95
BP121	How to Design and Make Your Own P.C.B.s	£1.95
BP122	Audio Amplifier Construction	£2.25
BP123	A Practical Introduction to Microprocessors	£1.95
BP124	Easy Add-on Projects for Spectrum, ZX81 & Ace	£2.75
BP125	25 Simple Amateur Band Aerials	£1.95
BP126	BASIC & PASCAL in Parallel	£1.50
BP127	How to Design Electronic Projects	£2.25
BP128	20 Programs for the ZX Spectrum and 16K ZX81	£1.95
BP129	An Introduction to Programming the ORIC-1	£1.95
BP130	Micro Interfacing Circuits – Book 1	£2.25
BP131	Micro Interfacing Circuits – Book 2	£2.25
BP132	25 Simple Shortwave Broadcast Band Aerials	£1.95
BP133	An Introduction to Programming the Dragon 32	£1.95
BP134	Easy Add-on Projects for Commodore 64, Vic-20, BBC Micro and Acorn Electron	£2.95
BP135	Secrets of the Commodore 64	£1.95
BP136	25 Simple Indoor and Window Aerials	£1.75
BP137	BASIC & FORTRAN in Parallel	£1.95
BP138	BASIC & FORTH in Parallel	£1.95
BP139	An Introduction to Programming the BBC Model B Micro	£1.95
BP140	Digital IC Equivalents and Pin Connections	£5.95
BP141	Linear IC Equivalents and Pin Connections	£5.95
BP142	An Introduction to Programming the Acorn Electron	£1.95
BP143	An Introduction to Programming the Atari 600/800 XL	£1.95
BP144	Further Practical Electronics Calculations and Formulae	£4.95
BP145	25 Simple Tropical and MW Band Aerials	£1.75
BP146	The Pre-BASIC Book	£2.95
BP147	An Introduction to 6502 Machine Code	£2.50
BP148	Computer Terminology Explained	£1.95
BP149	A Concise Introduction to the Language of BBC BASIC	£1.95
BP150	An Introduction to Programming the Sinclair QL	£1.95
BP152	An Introduction to Z80 Machine Code	£2.75
BP153	An Introduction to Programming the Amstrad CPC 464 and 664	£2.50
BP154	An Introduction to MSX BASIC	£2.50
BP155	International Radio Stations Guide	£2.95
BP156	An Introduction to QL Machine Code	£2.50
BP157	How to Write ZX Spectrum and Spectrum + Games Programs	£2.50
BP158	An Introduction to Programming the Commodore 16 and Plus 4	£2.50
BP159	How to Write Amstrad CPC 464 Games Programs	£2.50
BP161	Into the QL Archive	£2.50
BP162	Counting on QL Abacus	£2.50
BP163	Writing with QL Quill	£2.50
BP164	Drawing on QL Easel	£2.50
BP169	How to Get Your Computer Programs Running	£2.50
BP170	An Introduction to Computer Peripherals	£2.50
BP171	Easy Add-on Projects for Amstrad CPC 464, 664, 6128 and MSX Computers	£2.95
BP173	Computer Music Projects	£2.95
BP174	More Advanced Electronic Music Projects	£2.95
BP175	How to Write Word Game Programs for the Amstrad CPC 464, 664 and 6128	£2.95
BP176	A TV-DXers Handbook	£5.95
BP177	An Introduction to Computer Communications	£2.95
BP178	An Introduction to Computers in Radio	£2.95
BP179	Electronic Circuits for the Computer Control of Robots	£2.95
BP180	Computer Projects for Model Railways	£2.95
BP181	Getting the Most from Your Printer	£2.95
BP182	MIDI Projects	£2.95
BP183	An Introduction to CP/M	£2.95
BP184	An Introduction to 68000 Assembly Language	£2.95
BP185	Electronic Synthesiser Construction	£2.95
BP186	Walkie-Talkie Projects	£2.95
BP187	A Practical Reference Guide to Word Processing on the Amstrad PCW 8256 and PCW 8512	£5.95
BP188	Getting Started with BASIC and LOGO on the Amstrad PCW 8256 and PCW 8512	£6.95
BP189	Using Your Amstrad CPC Disc Drives	£2.95
BP190	More Advanced Electronic Security Projects	£2.95
BP191	Simple Applications of the Amstrad CPCs for Writers	£2.95
BP192	More Advanced Power Supply Projects	£2.95
BP193	Starting LOGO	£2.95
BP194	Modern Opto Device Projects	£2.95
BP195	An Introduction to Communications and Direct Broadcast Satellites	£3.95
BP196	BASIC & LOGO in Parallel	£2.95

BERNARD BABANI BP183

An Introduction to CP/M

- For the newcomer to computing, or even for someone who has experience with computers, it can be a little difficult at first to understand the basic function of an operating system such as CP/M.

- The primary purpose of an operating system is to enable the various parts, that make up a complete computer, work together in an orderly and co-ordinated way, and thus enable the computer to carry out its tasks efficiently and without crashing.

- CP/M is often called a 'disc operating system' which has led to the popular misconception that it is only concerned with the control and operation of disc drives. This is not the case, and while it is true that disc drives are very much at the centre of things, a full range of peripherals are controlled by CP/M.

- In order to run and use programs operating under CP/M it is not essential to have an understanding of the system, but a reasonable knowledge of the subject can certainly be of immense help when minor problems occur, and also in fully exploiting the possible potential of the system. This book tells the story!

ISBN 0-85934-157-7

00295

£2.95

9 780859 341578

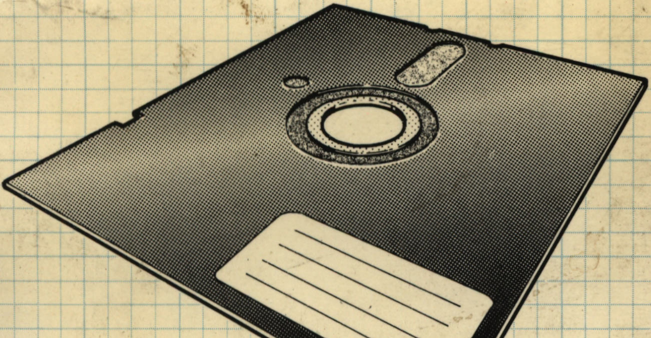
An Introduction to CP/M

BP183

R.A.PENFOLD

An Introduction to CP/M

R.A. PENFOLD



ALSO COVERS CP/M PLUS

BERNARD BABANI BP183

An Introduction to CP/M

- For the newcomer to computing, or even for someone who has experience with computers, it can be a little difficult at first to understand the basic function of an operating system such as CP/M.

- The primary purpose of an operating system is to enable the various parts, that make up a complete computer, work together in an orderly and co-ordinated way, and thus enable the computer to carry out its tasks efficiently and without crashing.

- CP/M is often called a 'disc operating system' which has led to the popular misconception that it is only concerned with the control and operation of disc drives. This is not the case, and while it is true that disc drives are very much at the centre of things, a full range of peripherals are controlled by CP/M.

- In order to run and use programs operating under CP/M it is not essential to have an understanding of the system, but a reasonable knowledge of the subject can certainly be of immense help when minor problems occur, and also in fully exploiting the possible potential of the system. This book tells the story!

ISBN 0-85934-157-7

00295

£2.95

9 780859 341578

AMSTRAD CPC

OCR

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, et non commerciales.
[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.
[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.